Optimal 2D Application Graphics

Session 216

These are confidential sessions—please refrain from filming or taking pictures.

Graphics and Media

Optimal 2D Application Graphics

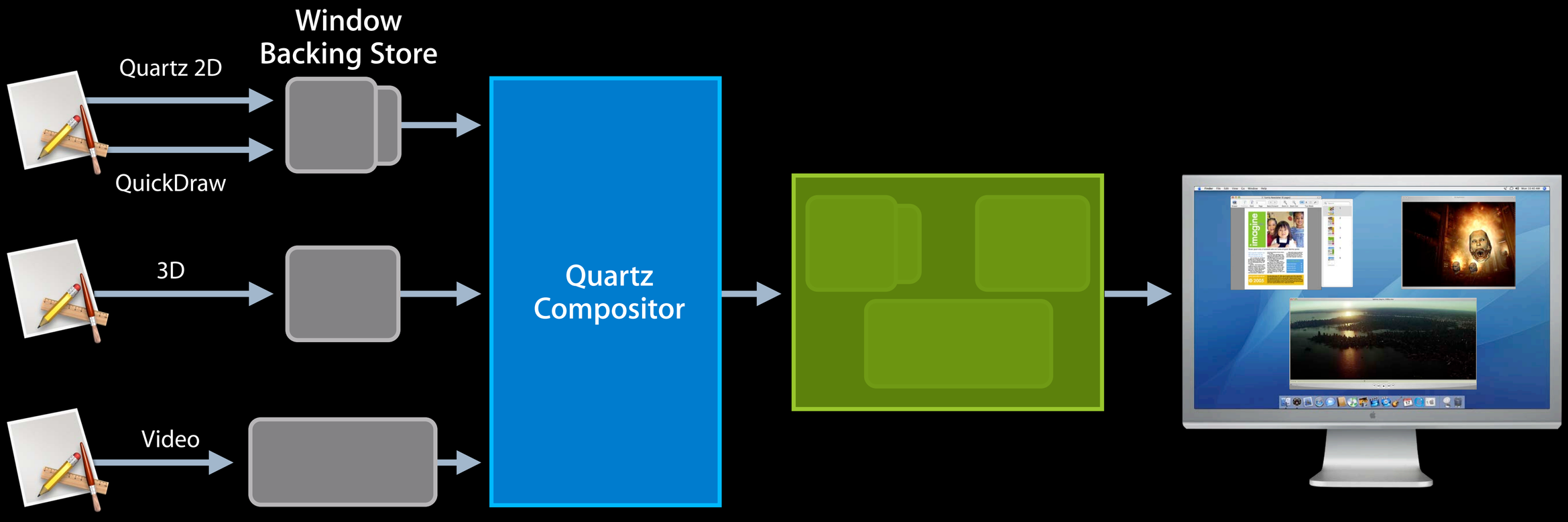
Andrew Barnes
Quartz Performance Guru
Feels the Need for Speed

http://developer.apple.com/wwdc2005/

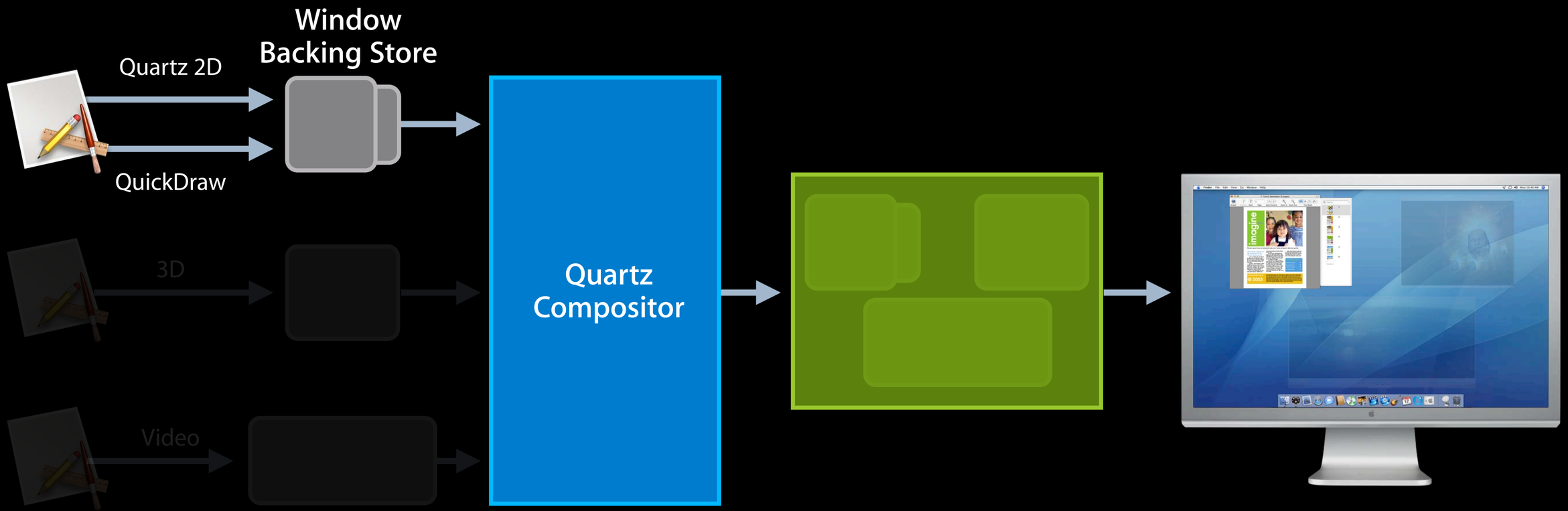
Introduction

- Efficient use of Mac OS X's graphics pipeline
- What you'll learn
  - Drawing optimizations
  - Optimized screen updates
  - Application graphics optimizations

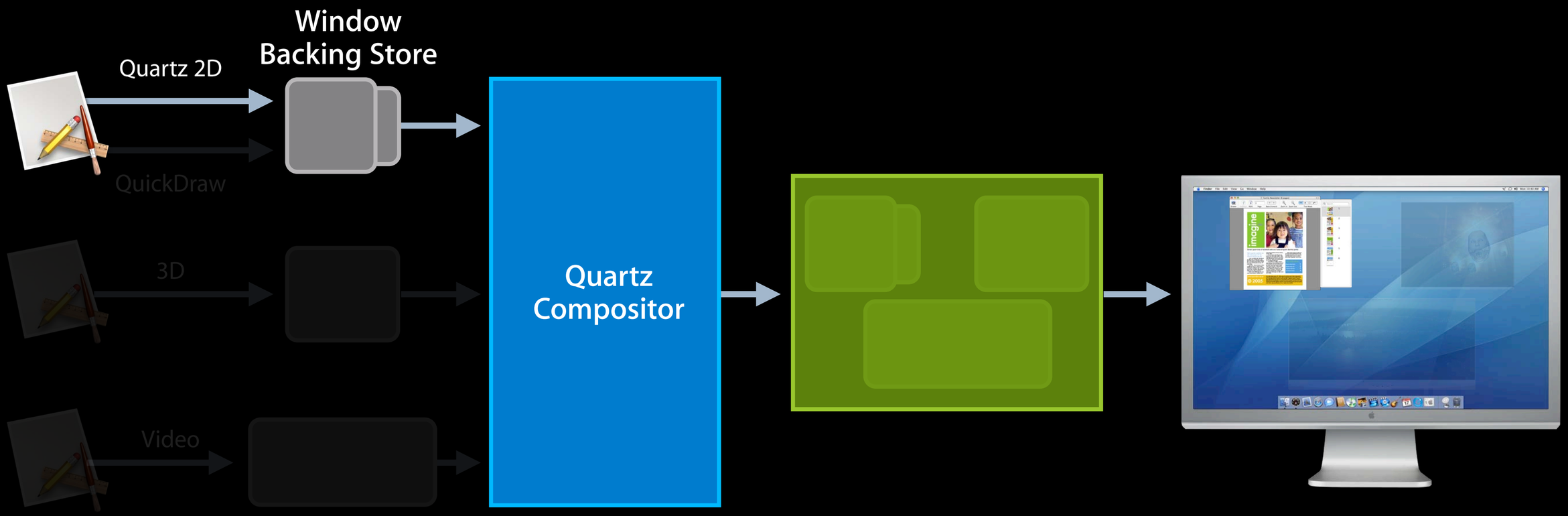
Graphics Architecture



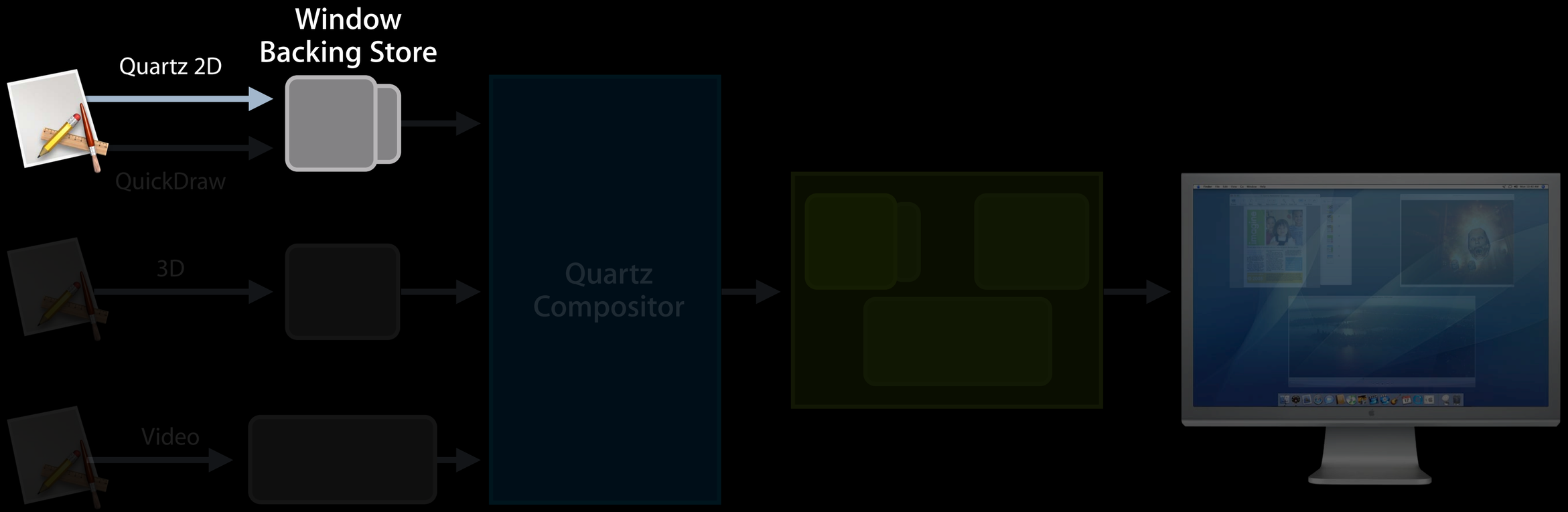
Graphics Architecture



Graphics Architecture



Graphics Architecture



About Quartz 2D Extreme

- OpenGL-based acceleration of the Quartz 2D API
- Final version not available in Tiger
- Lessons learned
  - For a majority of applications, rendering is not the bottleneck
  - Inefficient use of Quartz 2D API is even more costly for GPU
  - But, applications that use Quartz 2D efficiently win big!
- Developer action items
  - Test your application under Quartz 2D Extreme!
    - Debug version available in Tiger
  - Apply optimizations covered in this session!

Quartz Drawing Optimizations

Graphics Optimization Tenets

- Micro-tenets
  - Minimize state changes
  - Minimize drawing
  - Be cognizant of memory consumption
- Macro-tenets
  - Know your data set
  - Re-examine architectural decisions

Reuse Your Objects

- Quartz manages a working set of your objects
  - Your Refs are used as keys to internal caches
  - Silently improves performance of your application
  - Facilitates more efficient PDF/PS representations
- Release your Refs appropriately
  - Releasing your Refs will release any associated cached data

Types of Objects

Coverage Objects

- Paths
- Clips
- Text

Colorized Objects

- Colors
- Patterns
- Shadings
- Images
- Layers

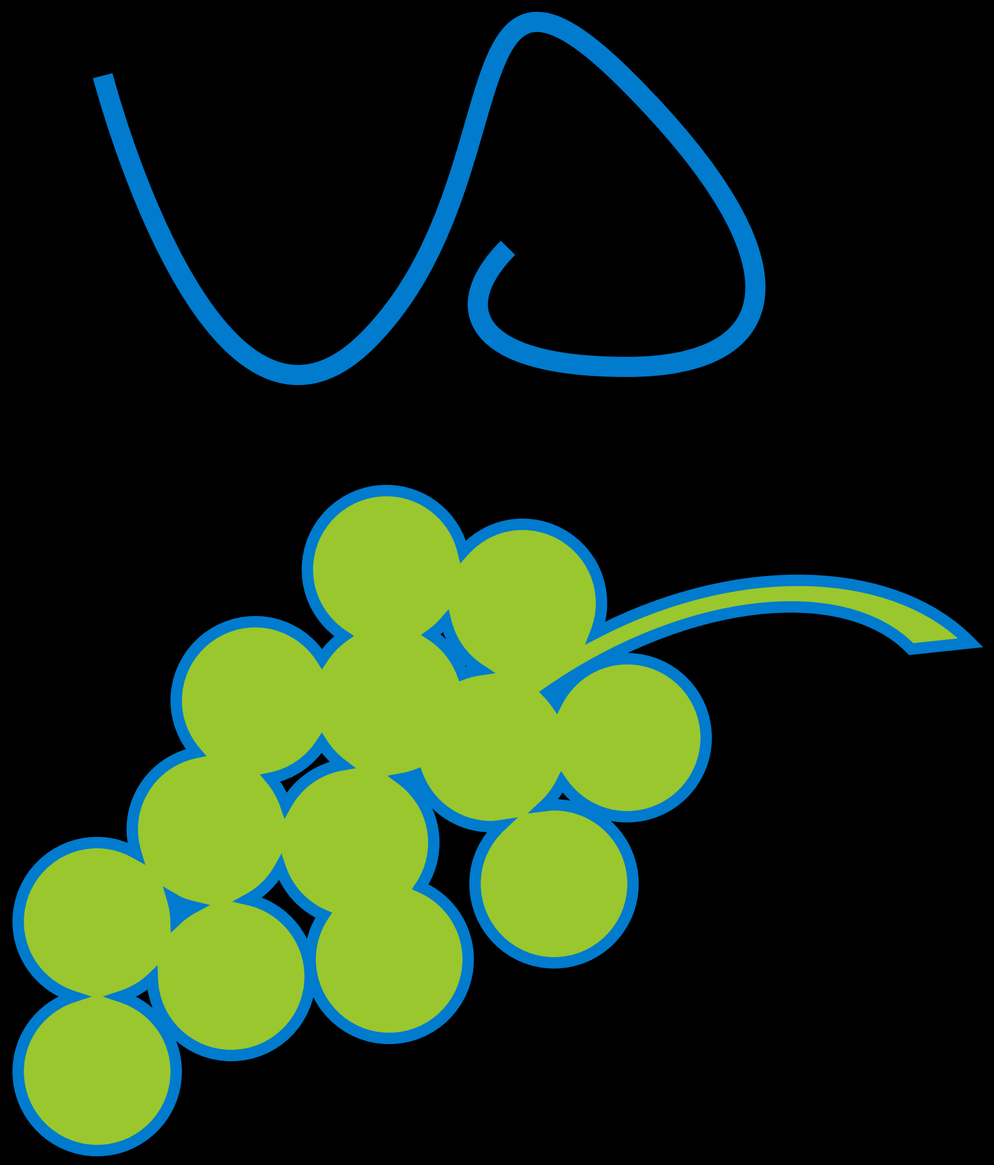
Types of Objects

Coverage Objects

- Paths
- Clips
- Text

Colorized Objects

- Colors
- Patterns
- Shadings
- Images
- Layers

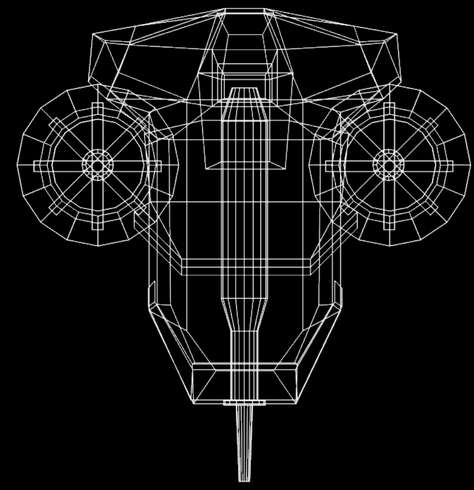


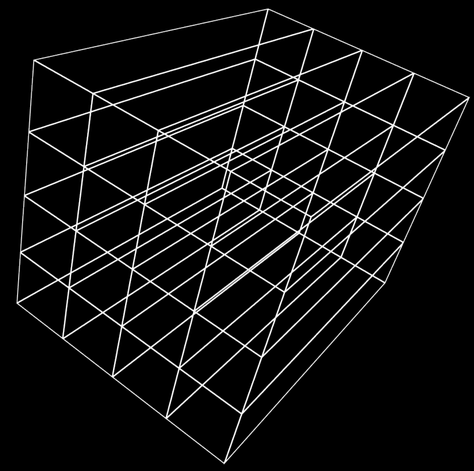
Paths

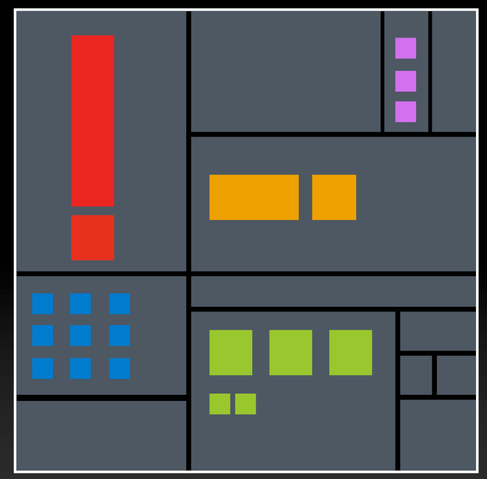
- `CGPathRefs` are general containers for arbitrary geometry
- Represents atomic coverage (i.e. each pixel is only touched once)
- `CGPathRefs` are always faster than path reconstruction
- Use path construction convenience APIs
- Avoid stroke state changes
  - Line-width, cap, join and dashes
- Know your geometric locality

Paths: Lines and Rectangles

- Use `CGContextStrokeLineSegments` and `CGContextDrawRects`
- Fastest way to draw lines and rectangles in Quartz
- Know your data set
  - Don’t draw objects that will not be seen, or are too small to see
  - Reorganize your data structures appropriately
    - Grids, Quadtrees, RTrees, etc.







Demo

Path and Lines

http://developer.apple.com/wwdc2005/

Clipping Tips

- Minimize your drawing
  - Do trivial culling of your objects
  - Do trivial clipping and/or use Quartz clipping
- Minimize your clipping
  - Only clip what really needs to be clipped
- Avoid clip state changes

Conference Breaks New Ground

By **Nina Lees**

ANTHONY Ranieri had no idea how he was going to support his young family when he was made redundant five years ago.

"When you're unemployed you go through a lot of stress and anxiety and you start to disbelieve your abilities," he said.

With a mortgage and family to support, he knew he had to find work — and fast.

He learned new job searching skills and within a few weeks was re-employed.

Seven months ago the Melbourne-based man began work with Ballarat's Haymes Paint as the human resources manager.

Now Mr Ranieri has used his job-searching techniques and his human resource skills to write a book to help others.

How to Find a Job in 6 Weeks provides information and advice from writing resumes and surviving panel interviews to tapping into the hidden job market.

"(Writing) the book changed my life and helped me to refocus. I had a mission to give something back to the community and teach people the tips and tricks of securing a job," Mr Ranieri said.

He has worked in a variety of contract management positions in the human resources area but will never forget the angst he felt while unemployed.

He said the five key factors in finding a job were:

- Successful job search skills.
- Strong financial management initiatives.
- Flexible and open career re-assessment.
- Mental toughness and a strong belief system.
- Managing your current job.

How to Find a Job in 6 Weeks will soon be available in bookstores and job networks.

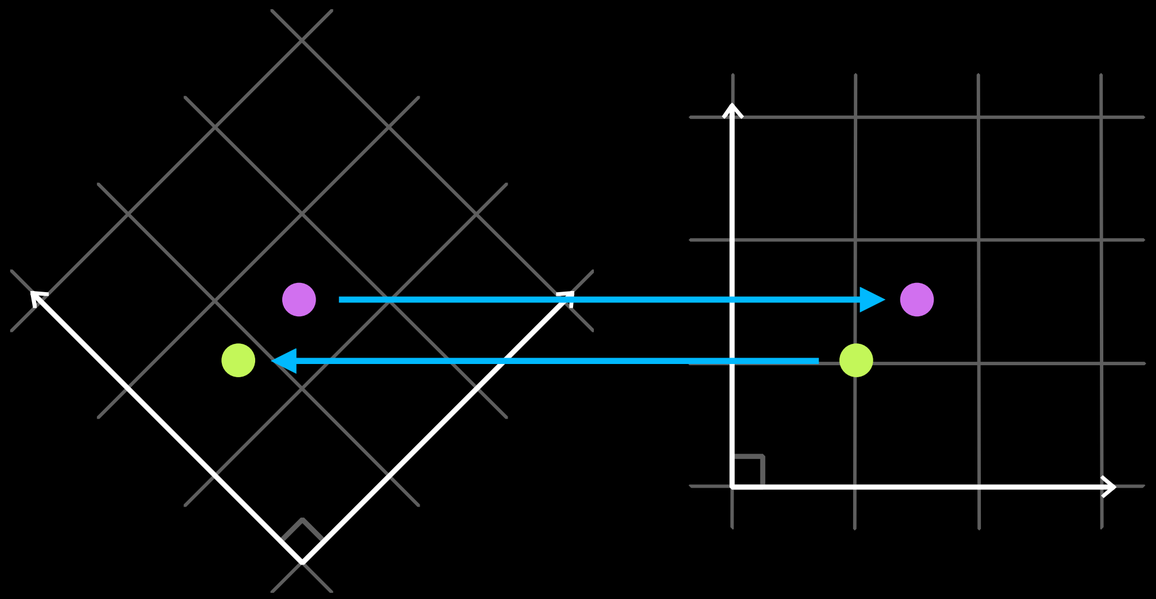
The 102-page book costs $35.95rrp and can be ordered at www.jobweb.com.au

IMAGE: *Herds of Developers flock to the Moscone Center West in San Francisco to gather the latest tools from Apple. Major annoucements were made in last weeks' World Wide Developer's Conference.*

More Clipping Tips

Clip on pixel boundaries

- Use `CGContextGetUserSpaceToDeviceSpaceTransform`
- Use convenience functions
  - `CGContextConvertPointToDeviceSpace` and `CGContextConvertPointToUserSpace`



More Clipping Tips

Image masks and clips

- Pre-render your geometry using an alpha only bitmap context
- Draw or clip to the resultant image mask
  - `CGContextDrawImage` or `CGContextClipToMask`

Text

- Quartz views text as a collection of paths
- Use ATSUI or Cocoa of text layout and/or rendering
- Cache your text layout appropriately
  - Layout is the most expensive part of text rendering
- Avoid text state changes
  - Font, size, text drawing mode, etc.

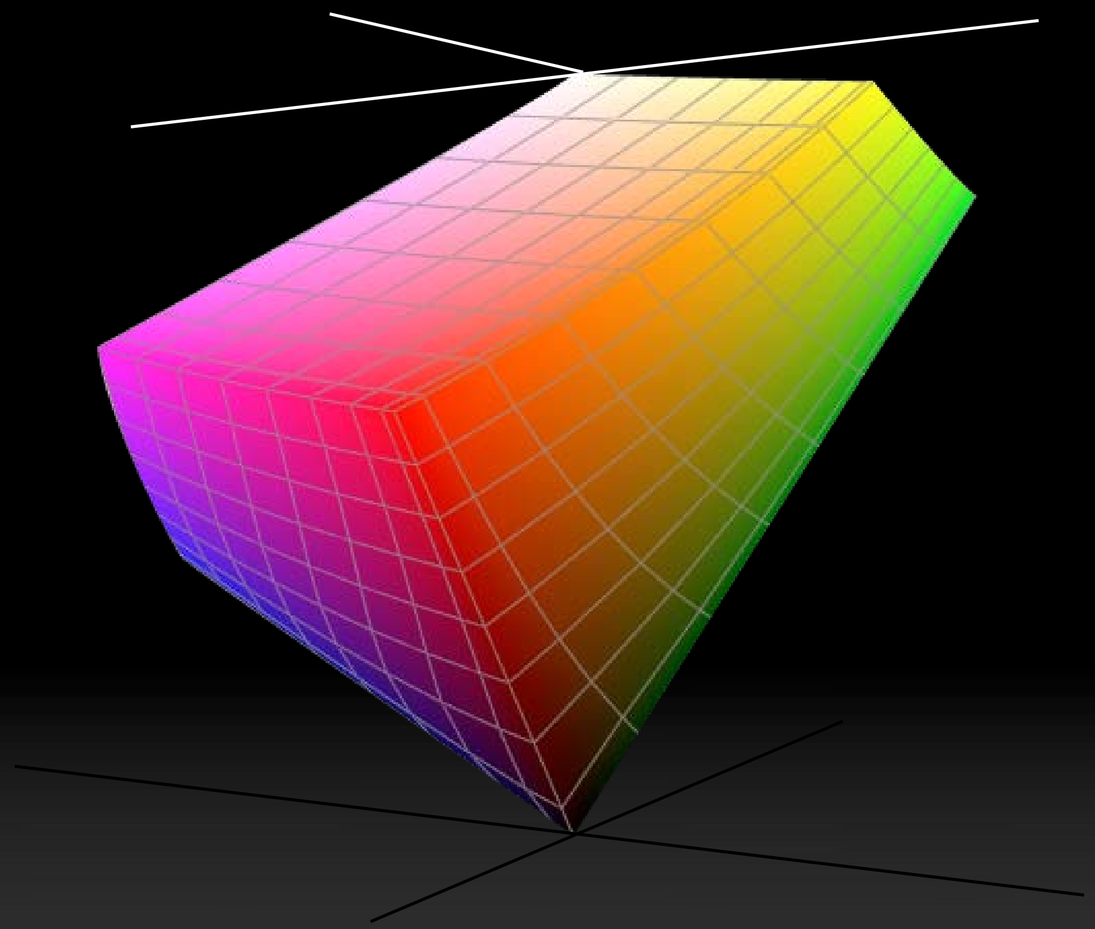
Types of Objects

Coverage Objects

- Paths
- Clips
- Text

Colorized Objects

- Colors
- Patterns
- Shadings
- Images
- Layers

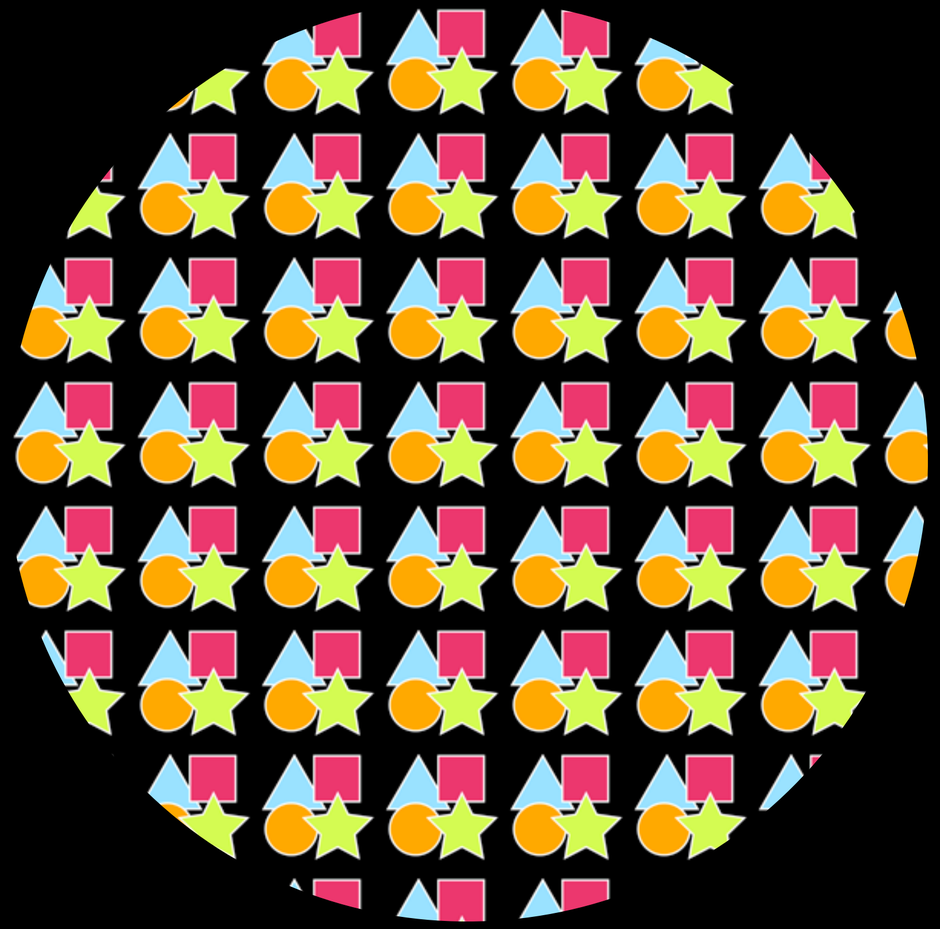


Colors

- `CGColorRefs` represent a single color or a pattern
- Setting color using `CGColorRefs` is always faster
- Use color spaces appropriately
  - Red=Green=Blue != Gray
- Avoid color state changes
- Use transparency effectively
- Use `CGContextSetAlpha` appropriately

Patterns

- `CGPatternRefs` represent tiled drawing
- Always faster than tiling yourself
- Patterns are cached by Quartz
- Use appropriate tiling modes and/or tile on pixel boundaries
  - `kCGPatternTilingConstantSpacing`



Shadings

- `CGShadingRefs` represent smooth transitions between colors
- Always faster than drawing individual colors
- Shadings are cached by Quartz
- `CGFunctionRefs` should be reused and implemented efficiently

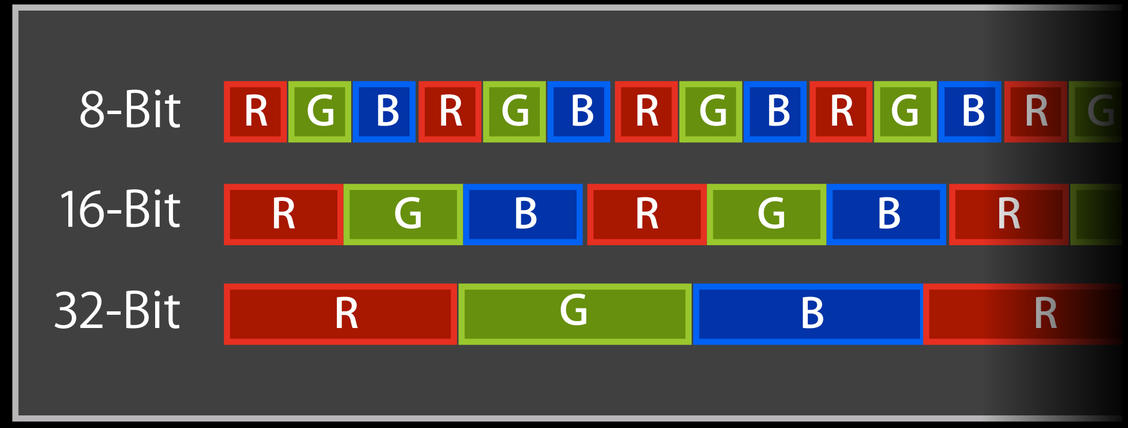
Images

- `CGImageRefs` are a general container for bitmap data
- Represents color and/or alpha data
- Images are cached by Quartz
- Use image interpolation quality appropriately
- Draw on pixel boundaries
- Use alpha effectively

Images

Know your pixel format

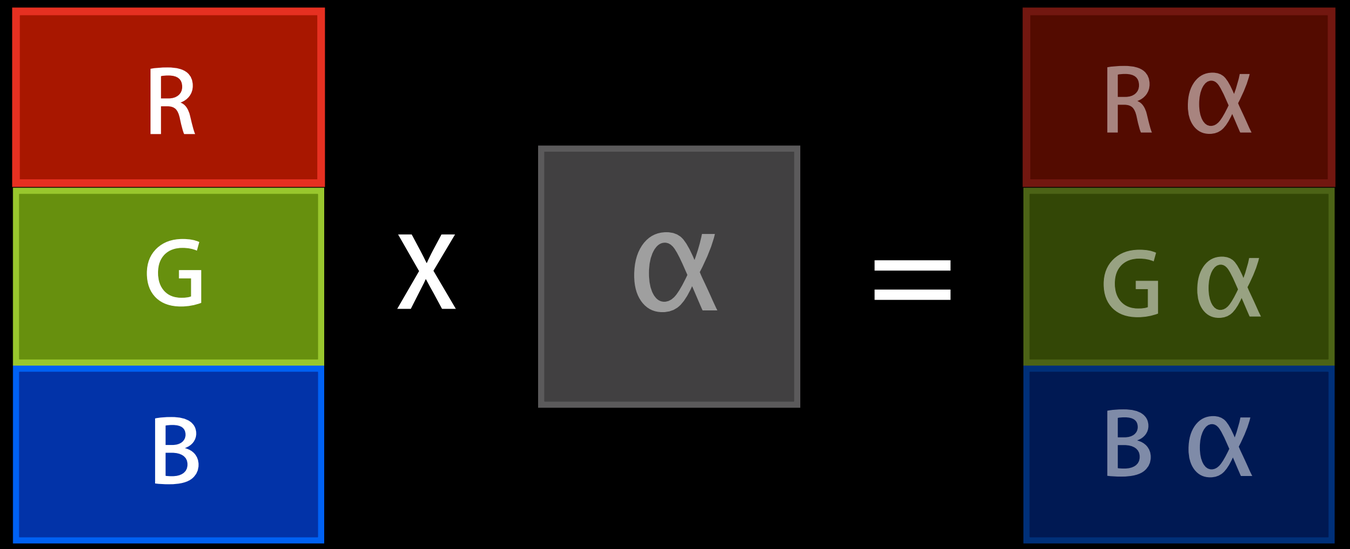
- Bytes-Per-Row alignment
  - Pixel multiples, vector, or cache line alignment
- Component size
  - Bytes, shorts, integers, and floats



Images

Know your pixel format

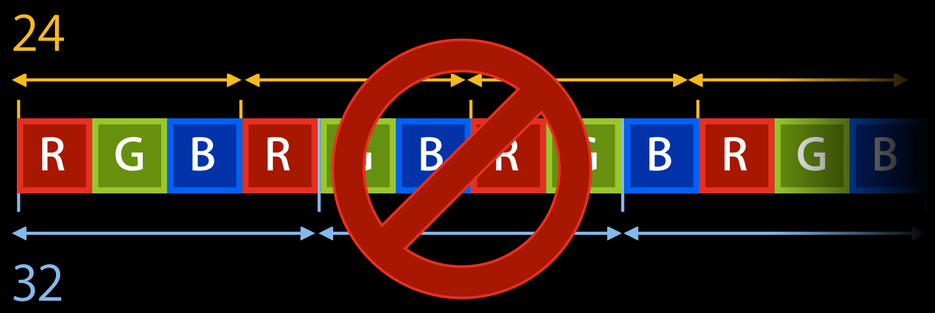
- Alpha pre-multiplication
  - Only if pre-multiplied data is more efficient for you to work with
  - Needless pre-multiplication may have performance penalties when color matching or interpolation is required



Images

Know your endianess

- Image data is defined as a network-order byte stream
- `CGBitmapInfo` byte-order constants
  - `kCGBitmapByteOrder16Big`, `kCGBitmapByteOrder32Big`
  - `kCGBitmapByteOrder16Little`, `kCGBitmapByteOrder32Little`
  - `kCGBitmapByteOrder16Host`, `kCGBitmapByteOrder32Host`
- Be prepared for image data with either endianess
  - Use `CGImageGetBitmapInfo`
- Use native for things that make sense



Images

Know your color space

- Use generic color spaces (`kCGColorSpaceGenericRGB`, etc.)
  - Least number of colorants for which you have data
  - Avoid needless coercion of your data
- For best imaging performance
  - Use the destination color space
  - Not always color managed
- Don’t know my destination color space?
  - Use `CGBitmapContextGetColorSpace` for bitmap contexts
  - Use the display profile for window contexts

Images
Caching

- Pre-render your image to a `CGBitmapContextRef`
  - Create a `CGBitmapContextRef` of appropriate size, color space, and format
  - Remember, use transparency effectively
  - Draw your image to the `CGBitmapContextRef`
- Create a `CGImageRef` from your `CGBitmapContextRef`
- Draw the resultant image

Images

Advanced caching

- Use data providers to your advantage
  - Data providers can be used to decouple your image caching behavior
  - Pre-rendered or pre-decompressed data can be constructed on demand and discarded appropriately

Images

I/O

- The master of all image formats
- The fastest decoders and encoders on the platform
- Lots of bells and whistles
  - Rich set of image formats
    - Web standards, HDR, Raw Camera, other…
  - Incremental loading and metadata support
  - Image manipulation
  - Much more to come
- Use decompression caching effectively

Layers

- `CGLayerRefs` are device dependent representations of your drawing
- Primarily used for capturing complex drawings
- `CGLayerRefs` are created from `CGContextRefs`
  - Hides complicated details about destination compatibility
- Use `CGLayerRef` instead of `CGBitmapContextRefs`

Demo

Imaging

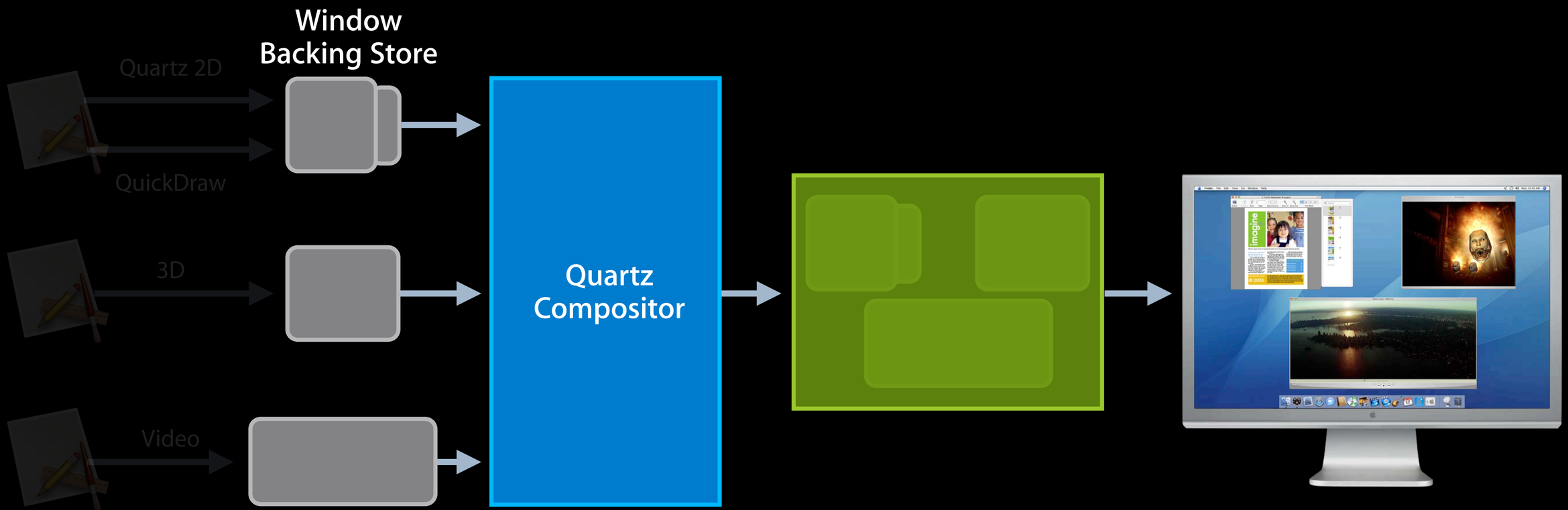
Haroon Sheikh

http://developer.apple.com/wwdc2005/

Optimized Screen Updates

http://developer.apple.com/wwdc2005/

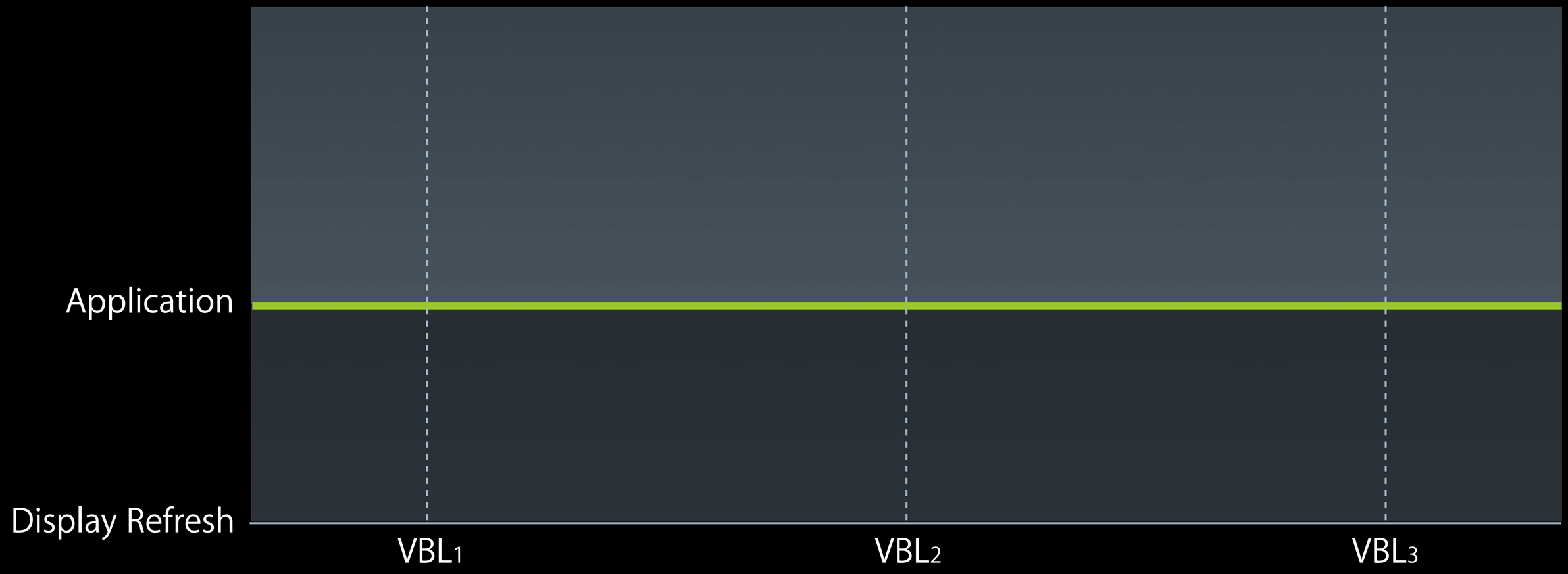
Graphics Architecture



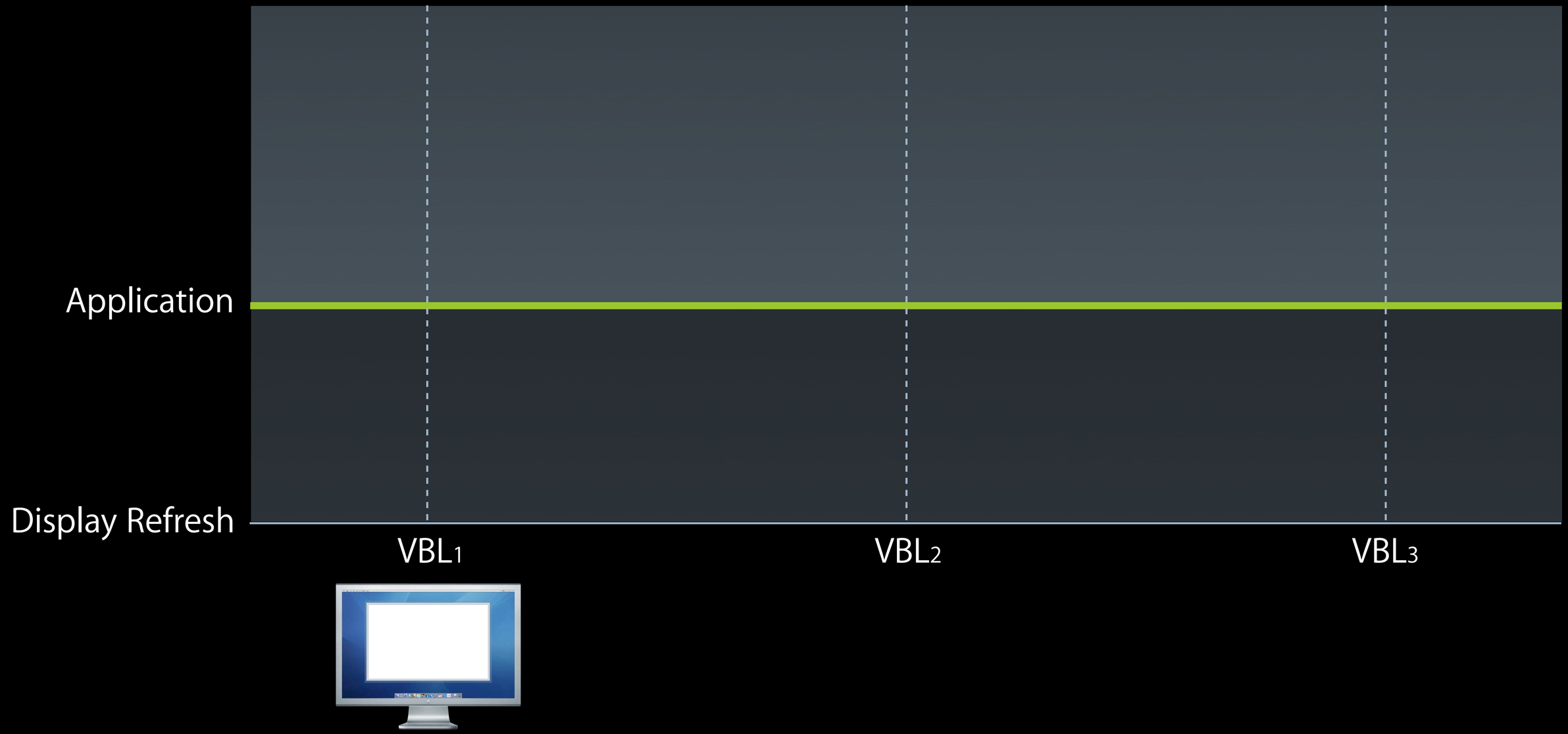
Flushing

- Flush only when the user needs to see what has been drawn
- Users can’t see anything flushed faster than refresh rate of the display
- Over-flushing only consumes valuable CPU and GPU resources

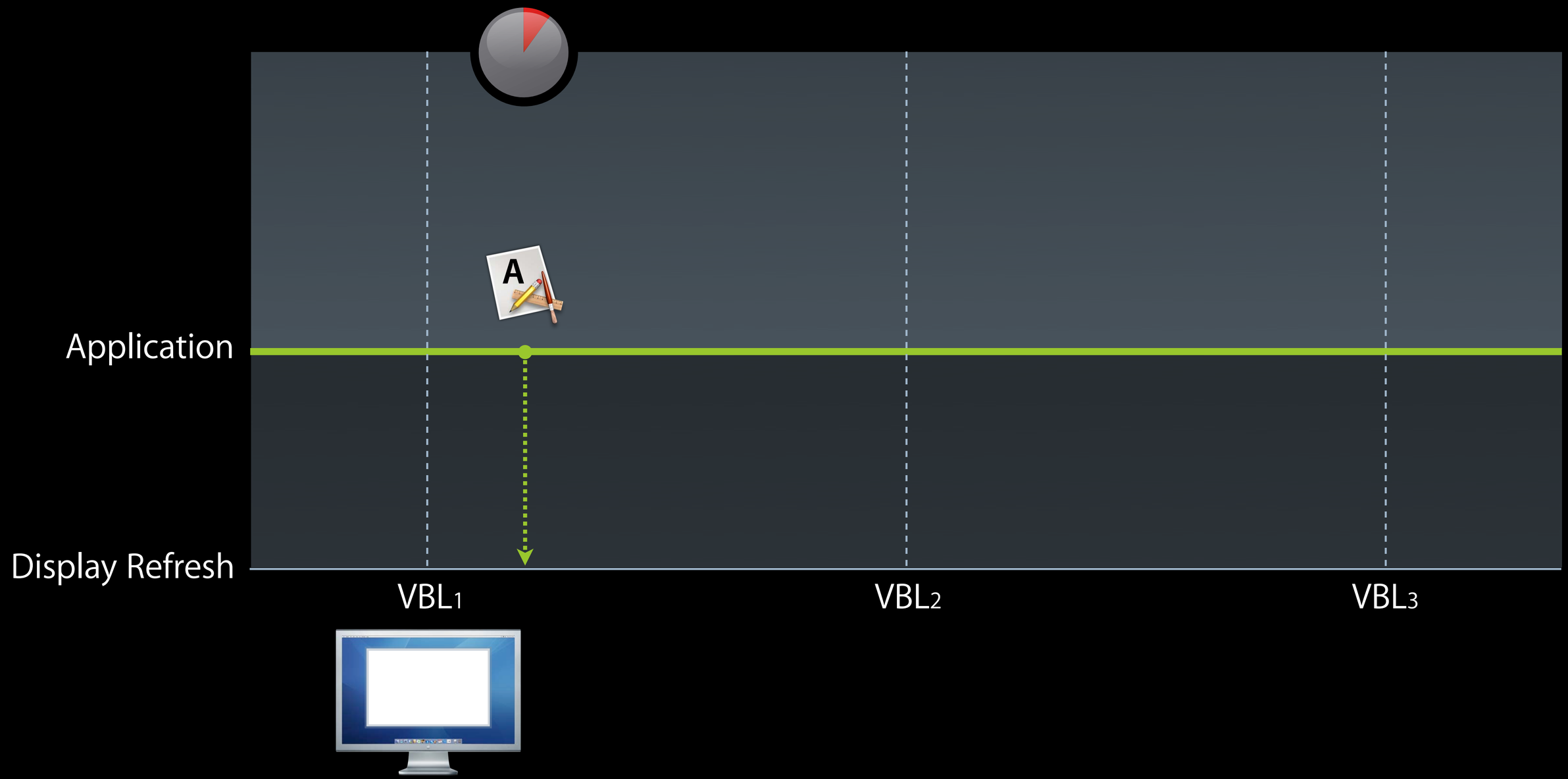
Flushing



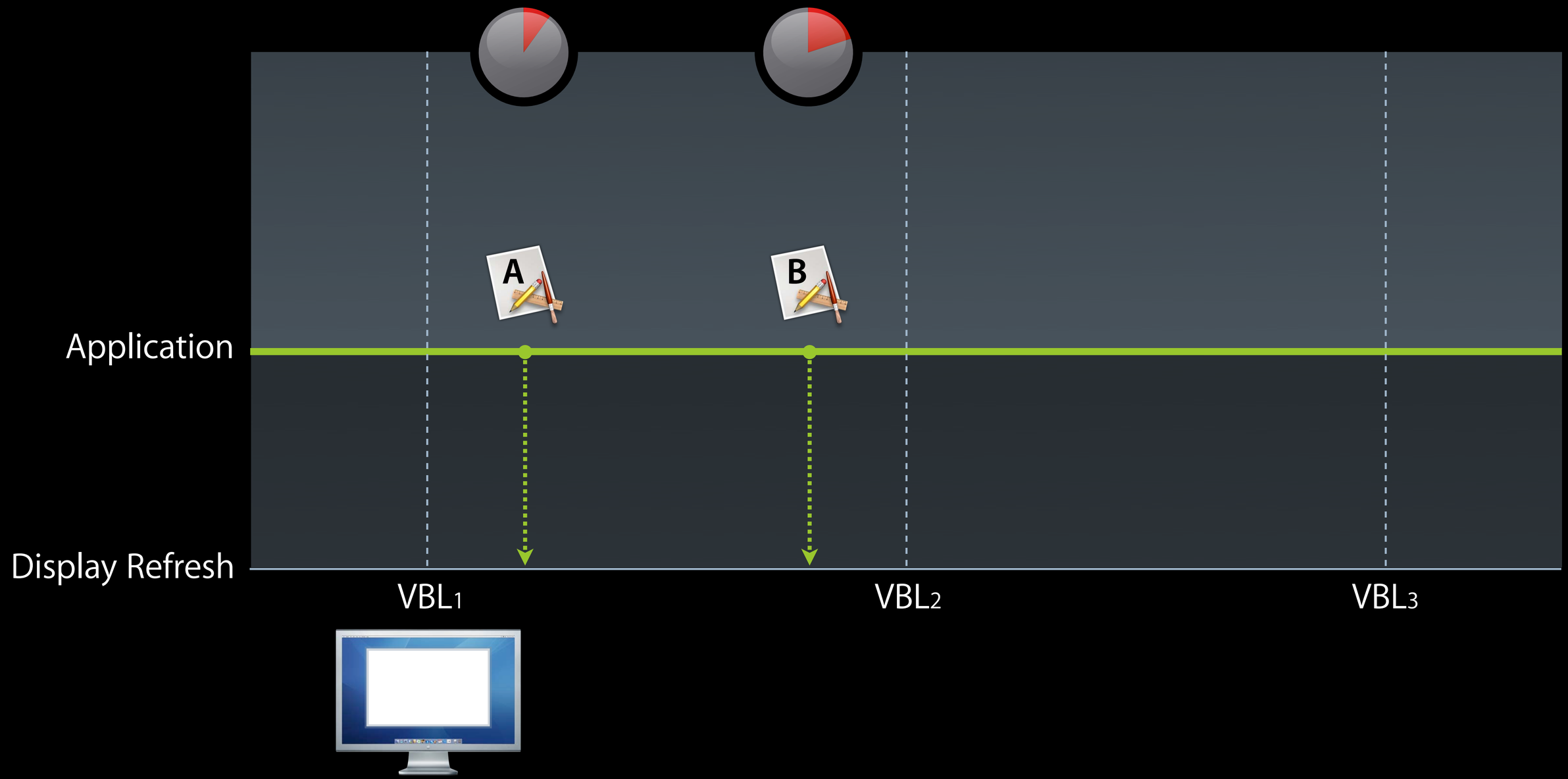
Flushing



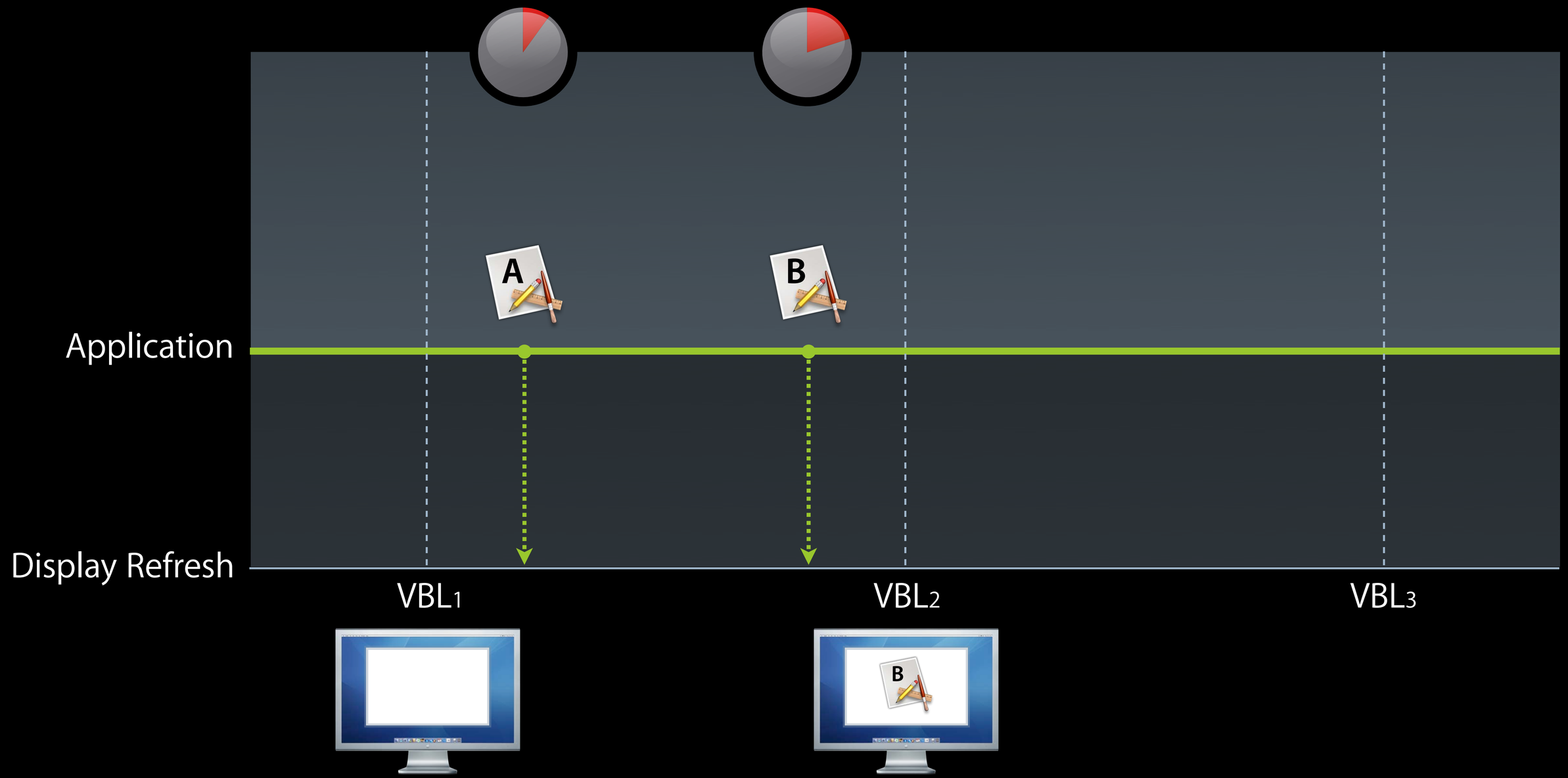
Flushing



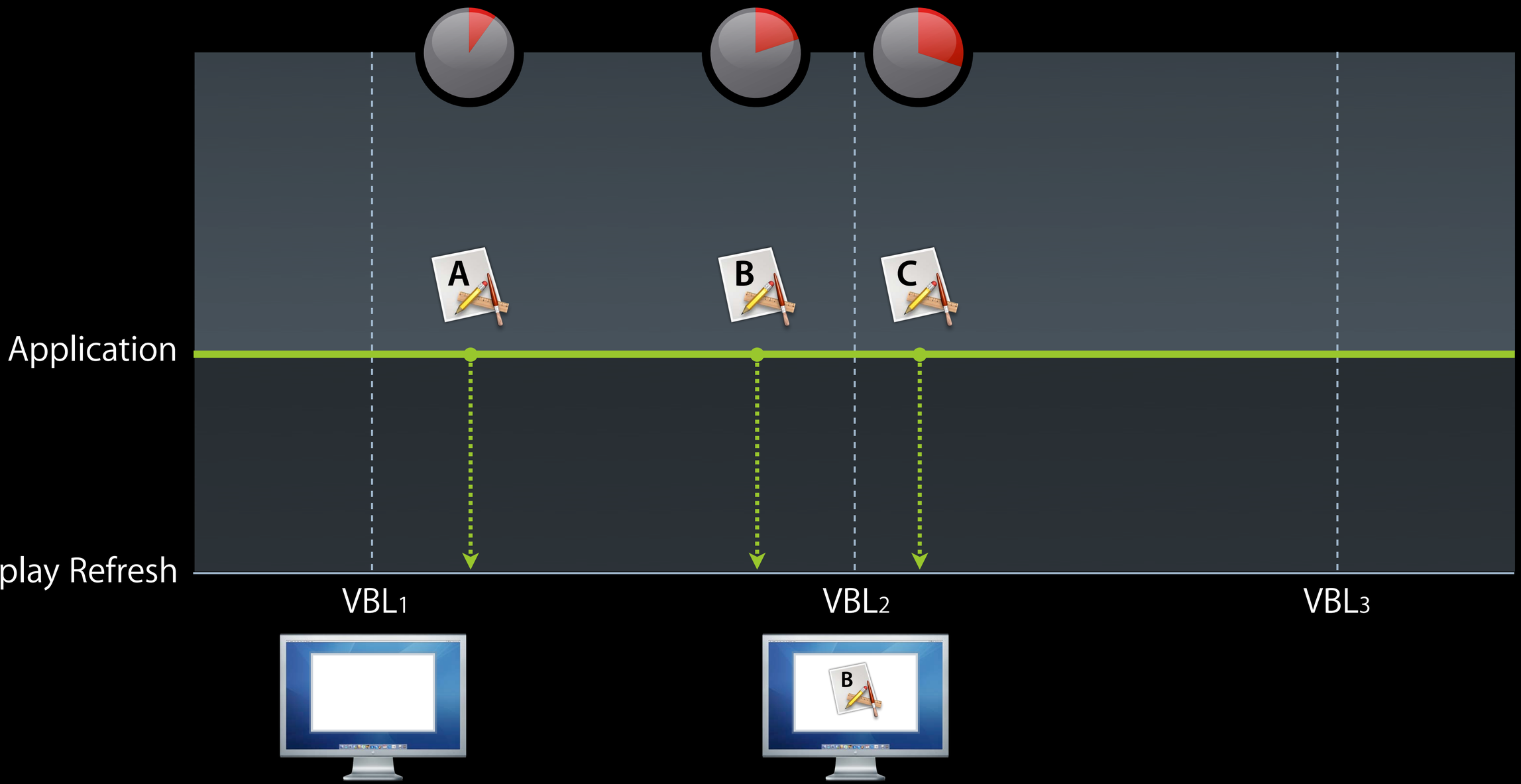
Flushing



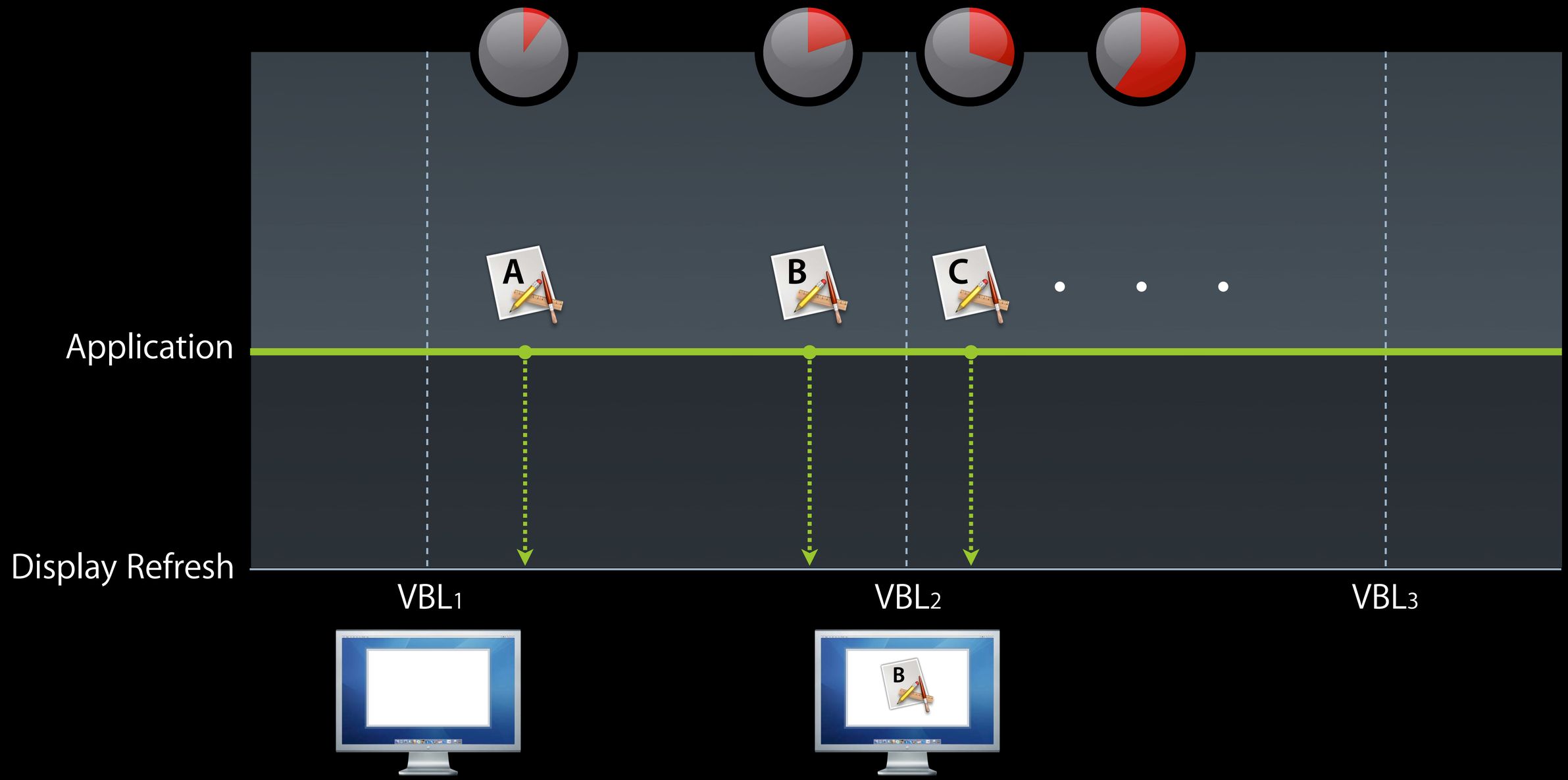
Flushing



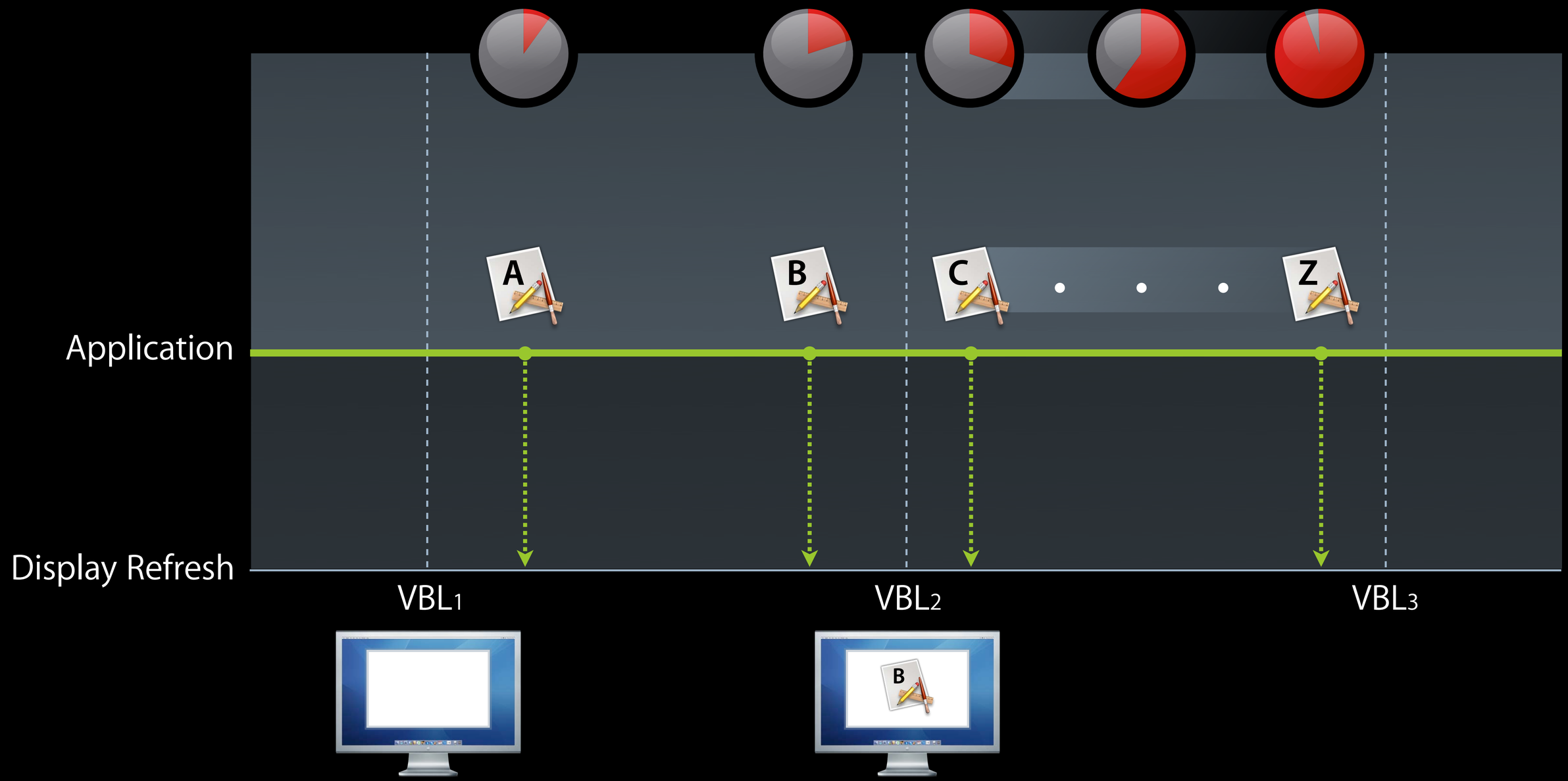
Flushing



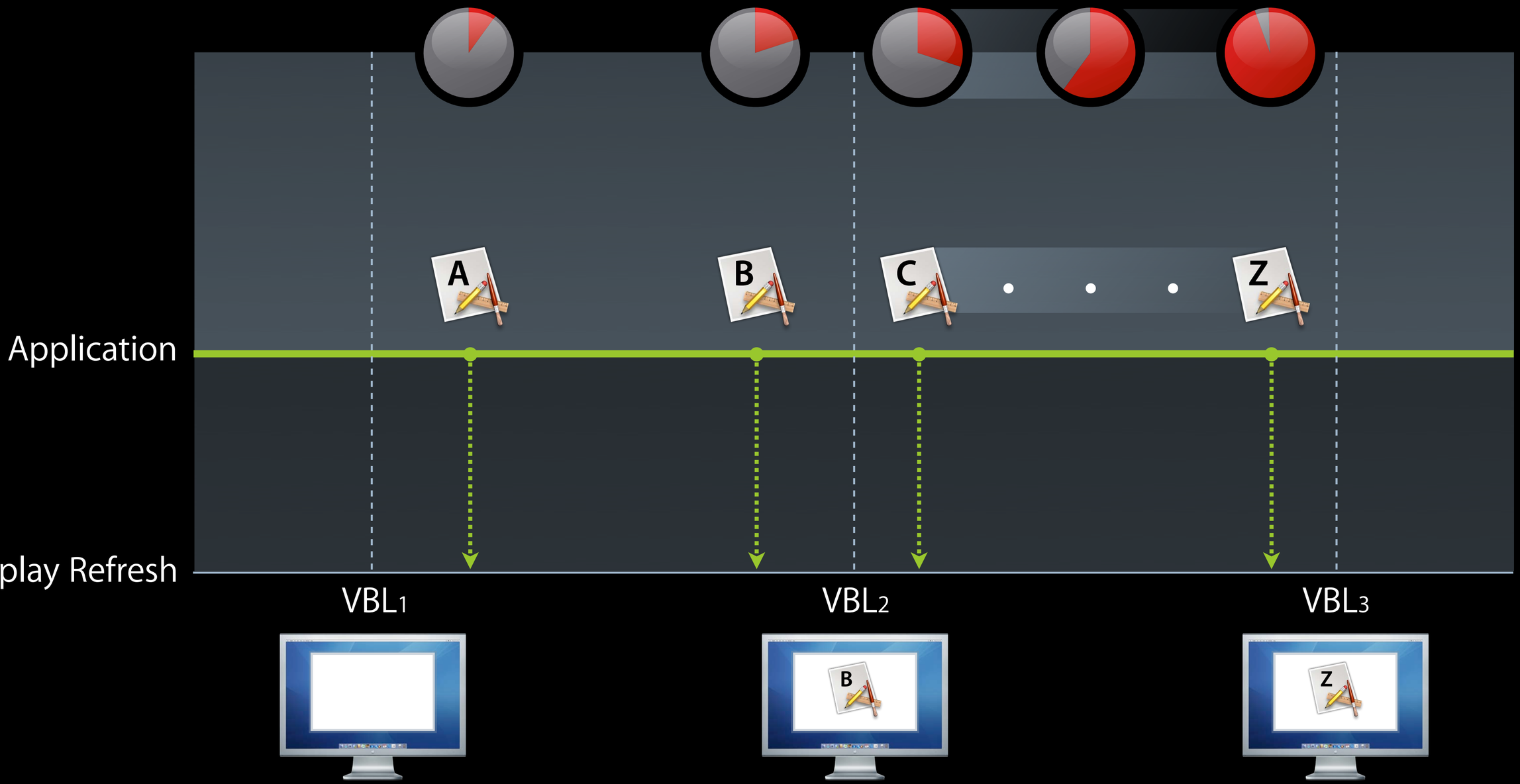
Flushing



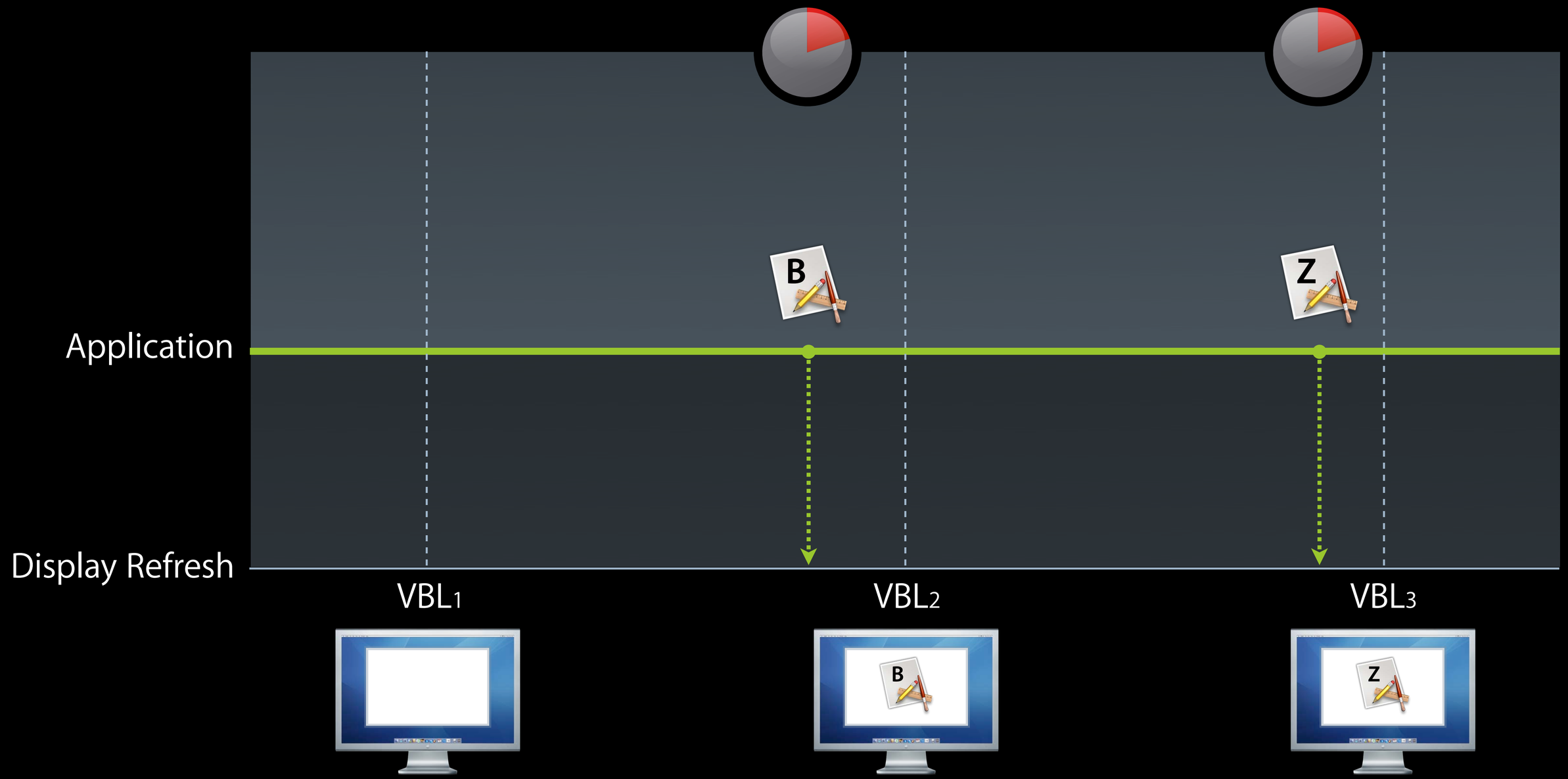
Flushing



Flushing



Flushing



Optimal Flushing

- Animated drawing should be flushed using a single periodic timer
  - Timer interval ~= 1/2 refresh rate of the display
  - Drawing time <= timer interval
- Decouple your visualization engine from your data engine
  - Neither engine should impede the other
  - Network and disk intensive applications will benefit from separate threads

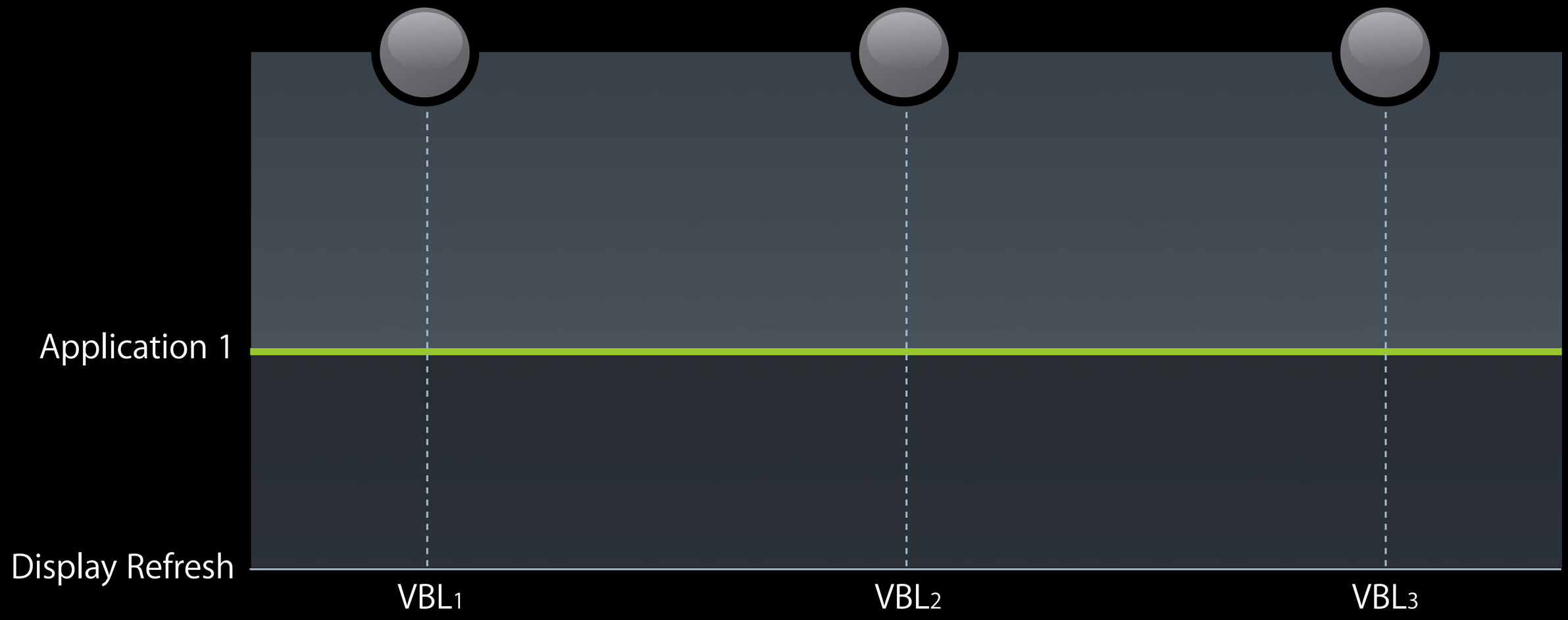
Demo

Worm

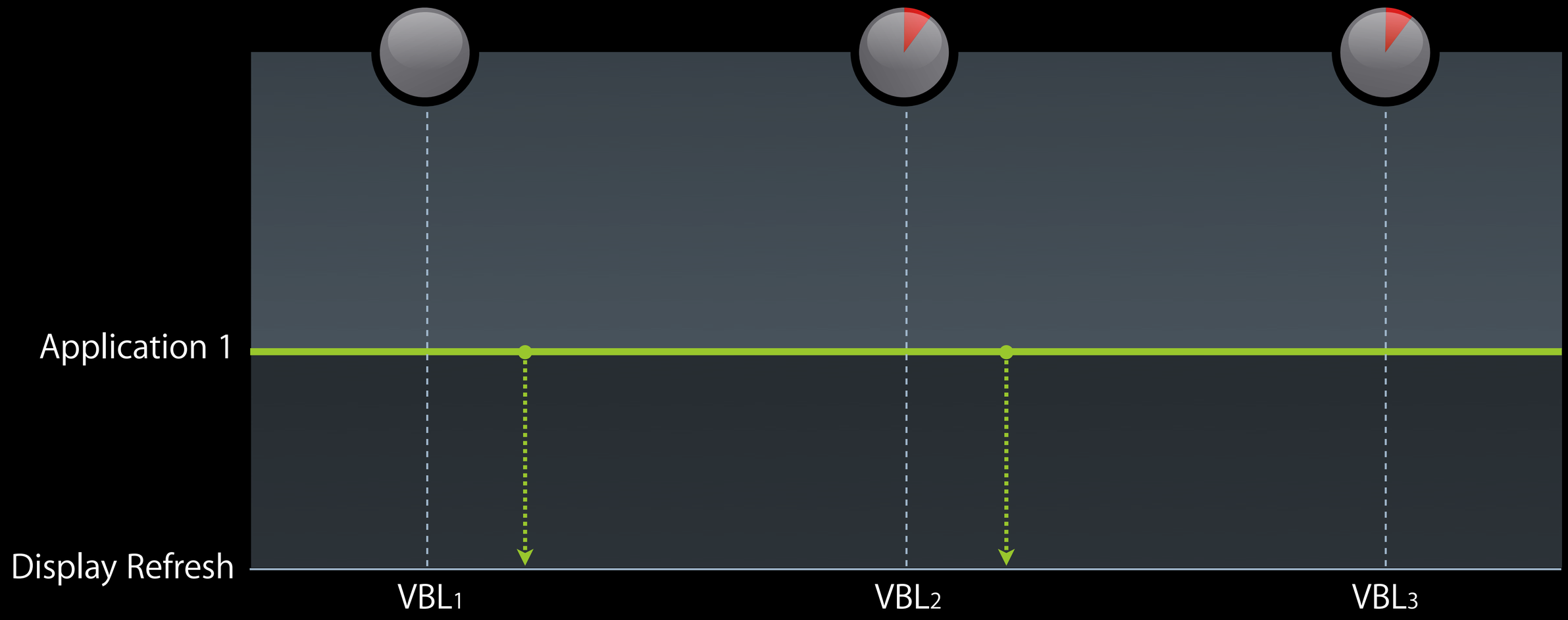
Haroon Sheikh

http://developer.apple.com/wwdc2005/

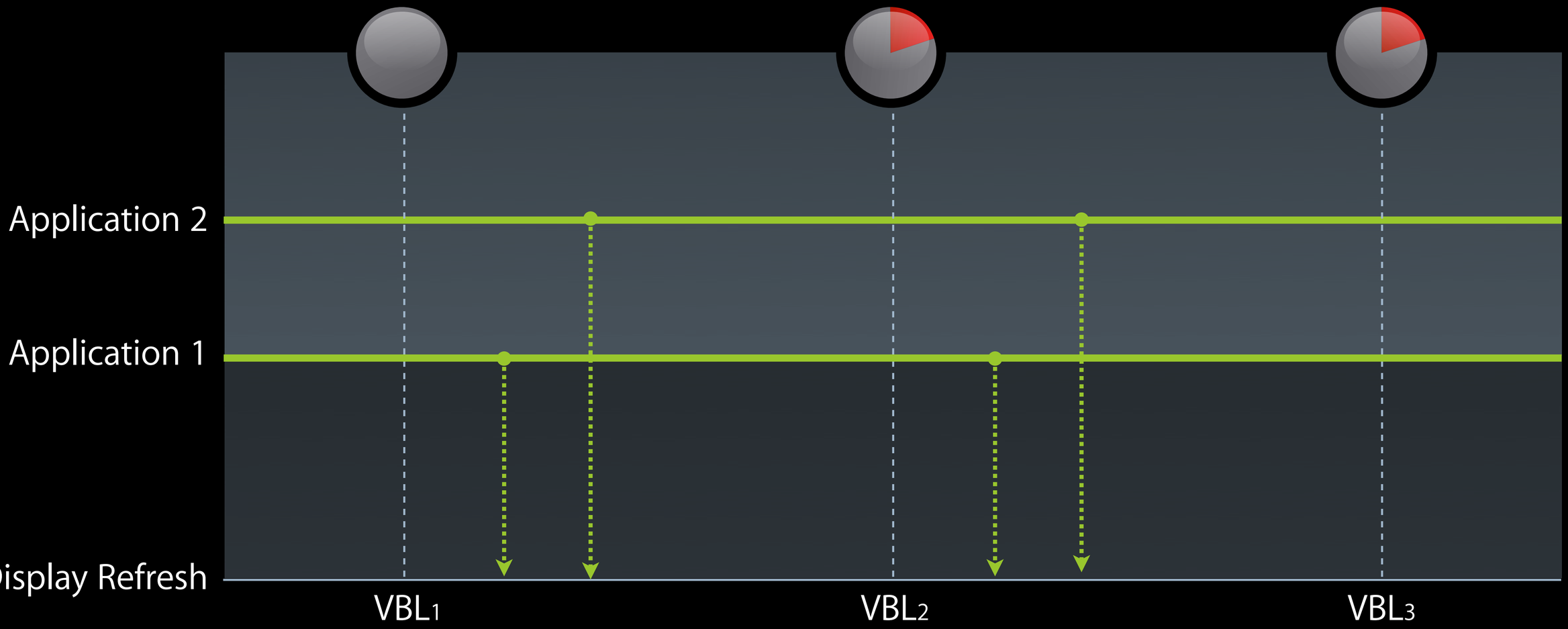
Updates



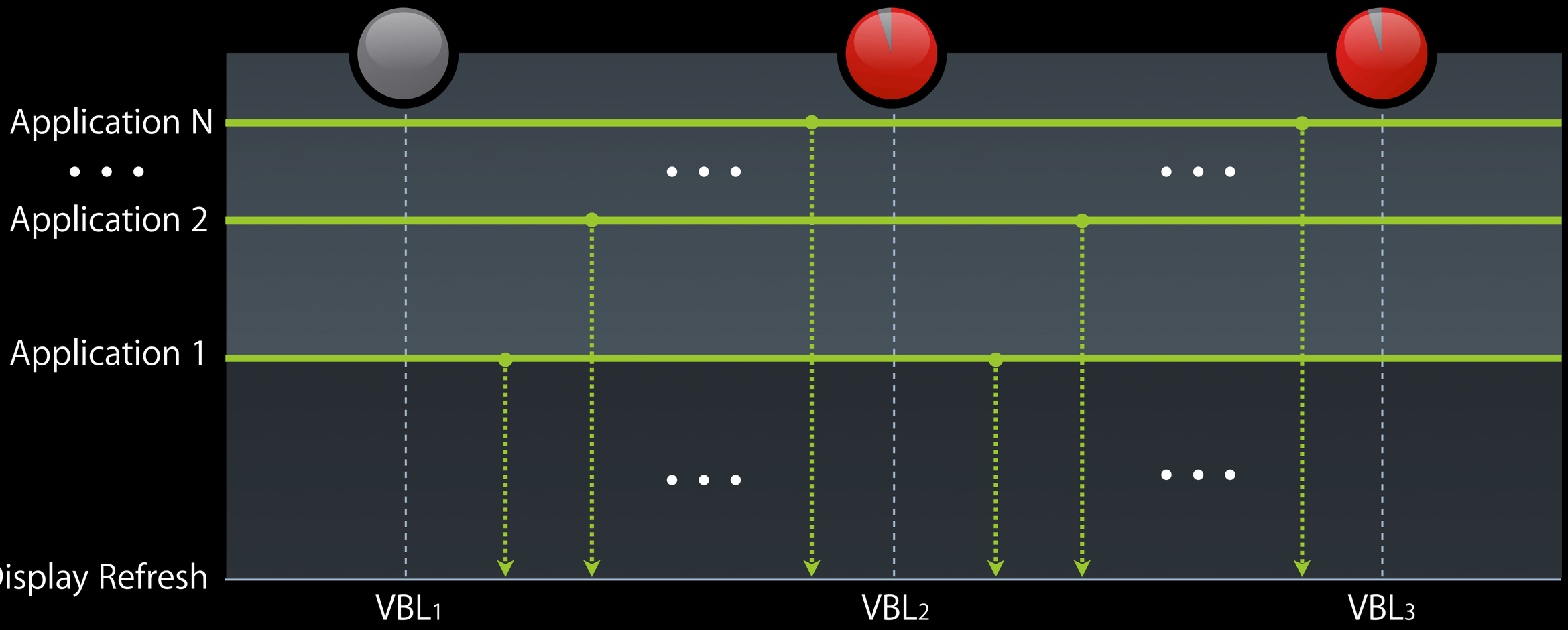
Updates



Updates


Application 2
Application 1
Display Refresh
VBL1
VBL2
VBL3

Updates


Application N
Application 2
Application 1
Display Refresh
VBL1
VBL2
VBL3

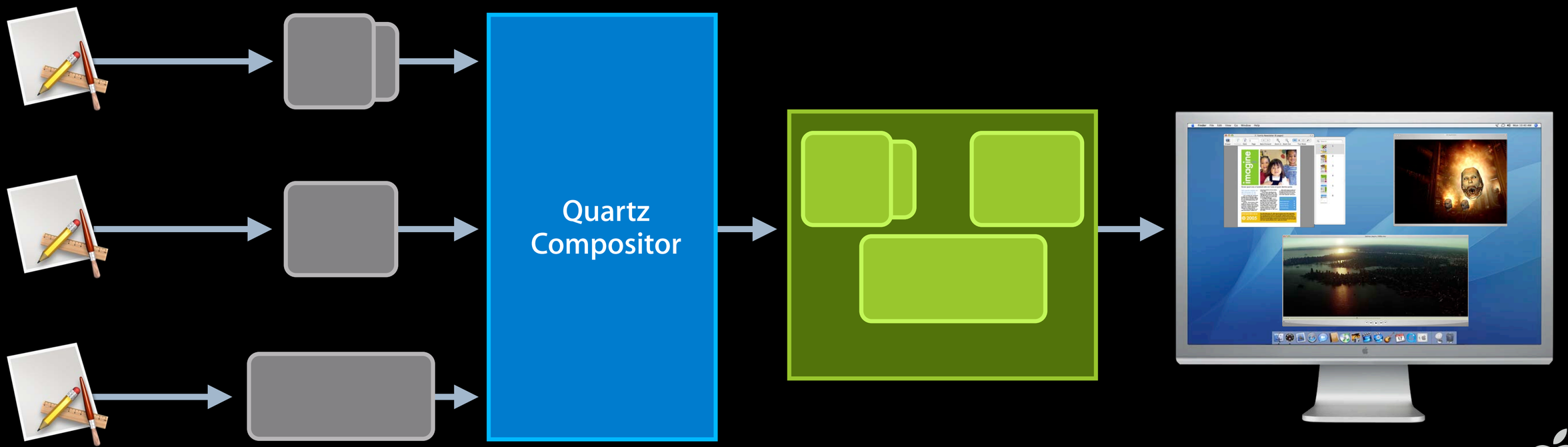
Demo

Multiple applications

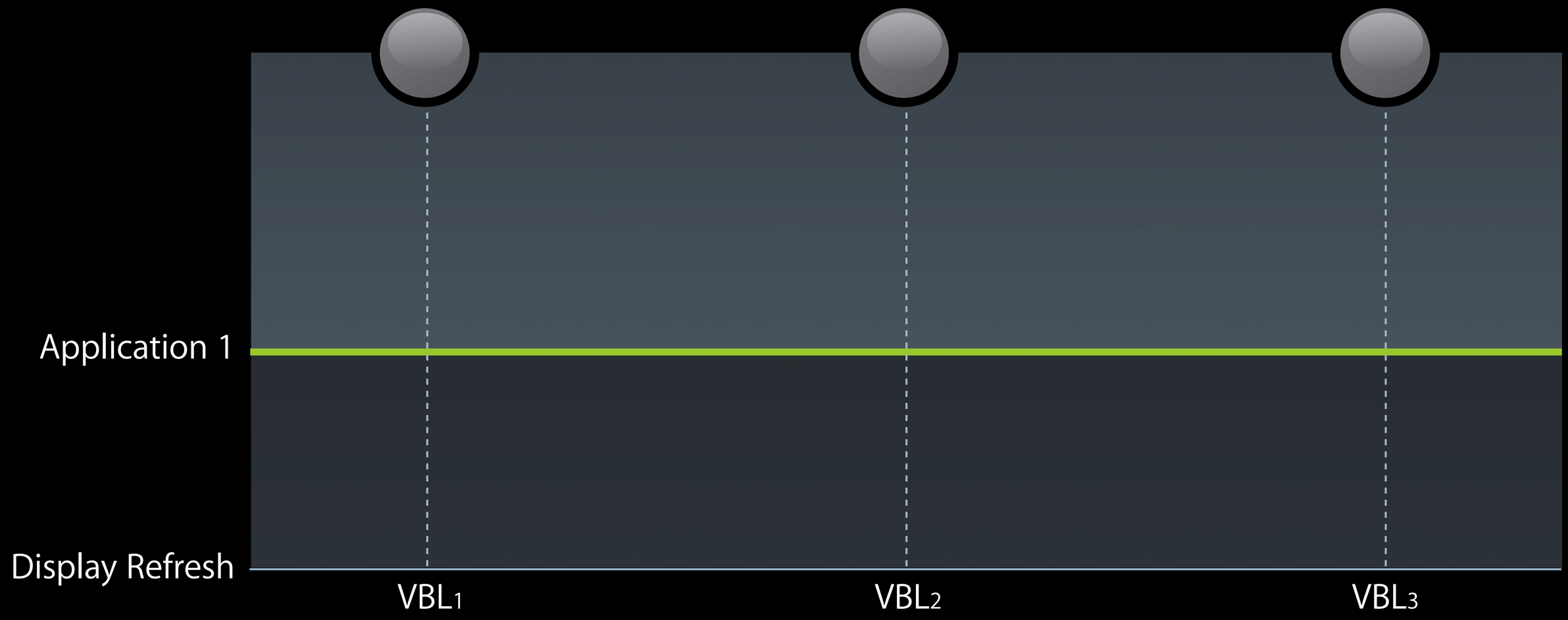
http://developer.apple.com/wwdc2005/

Coalesced Updates

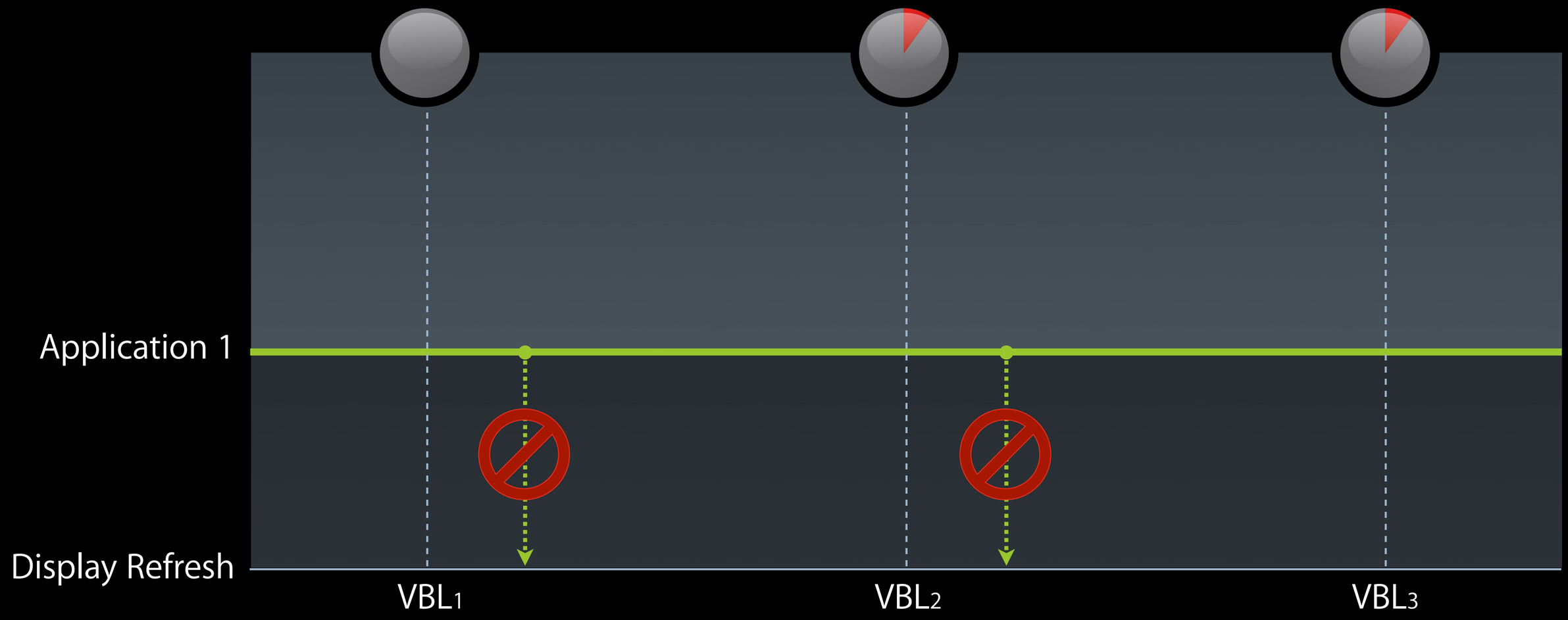
- The Quartz Compositor is the guardian of all updates to the display
- Flushes from all applications are coalesced into a single update
  - Updates are synchronized with the refresh rate of the display
- Coalescing updates makes better system-wide use of CPU and GPU
  - Especially true for simultaneous updates from many applications



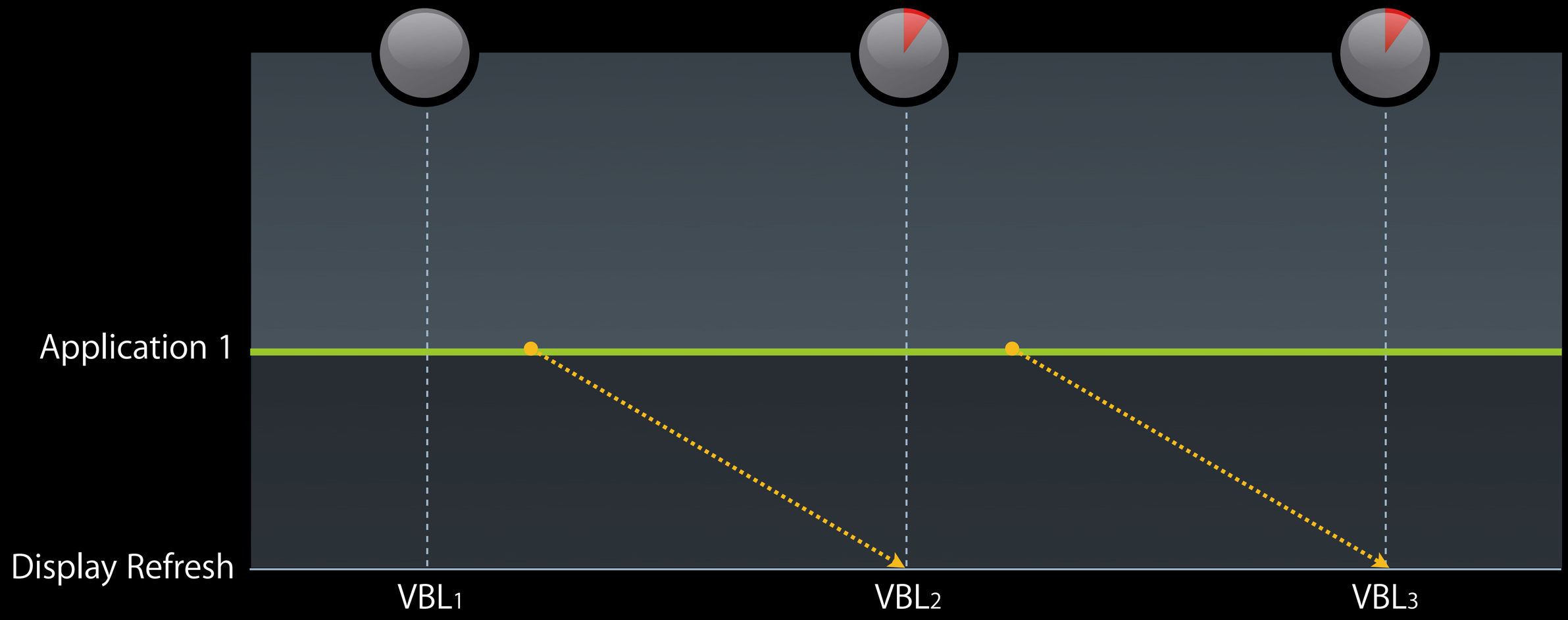
Coalesced Updates



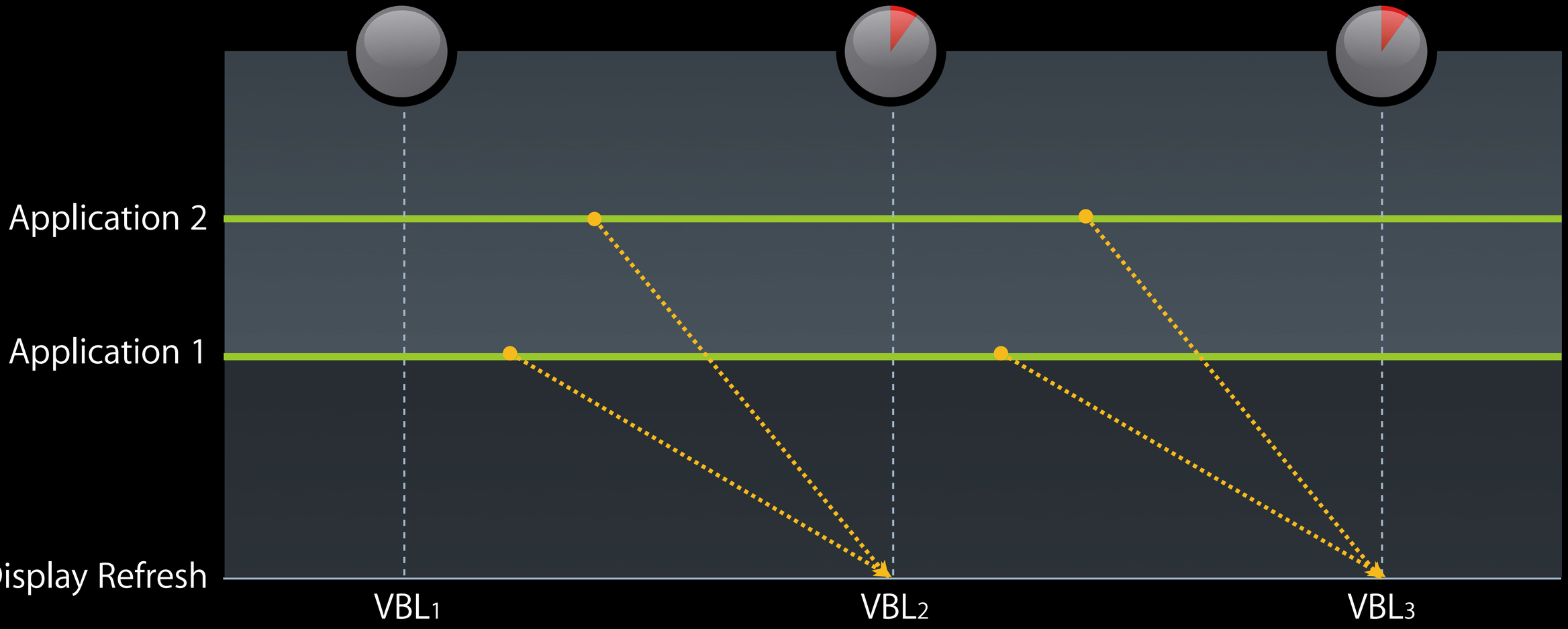
Coalesced Updates



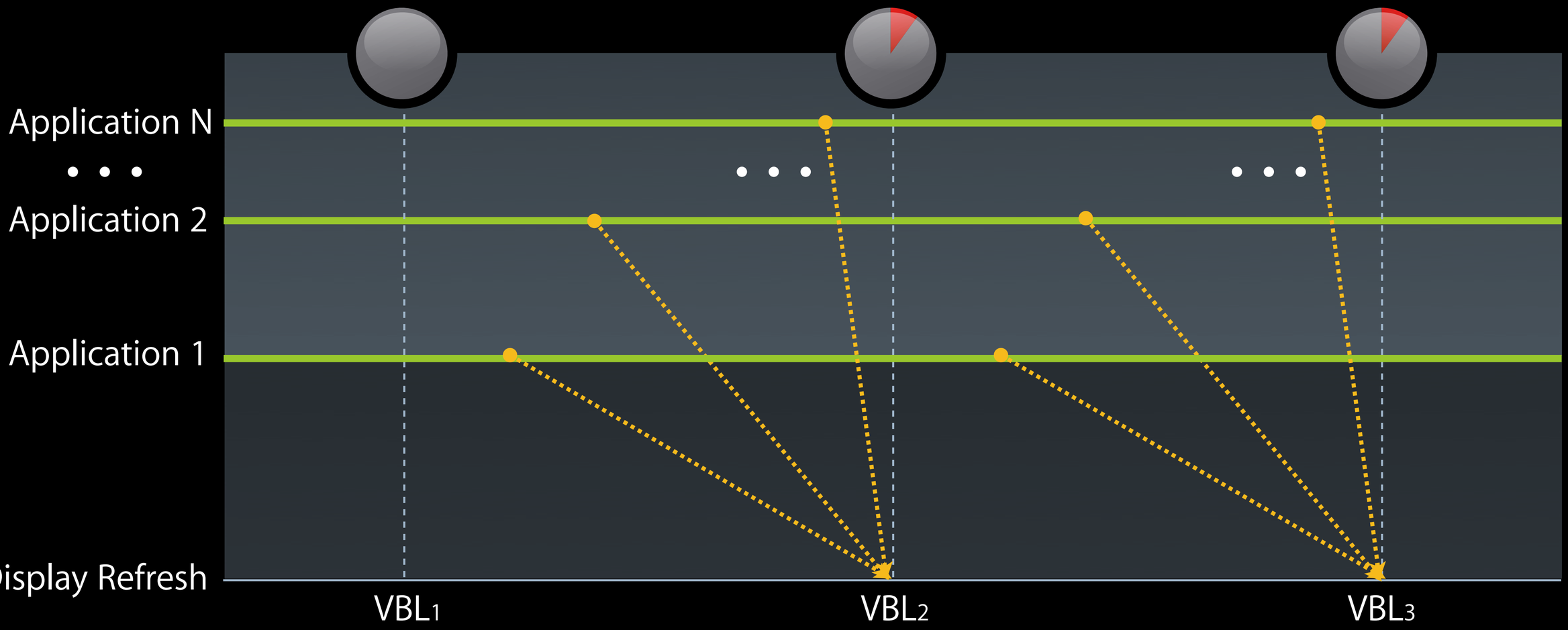
Coalesced Updates



Coalesced Updates



Coalesced Updates



Demo

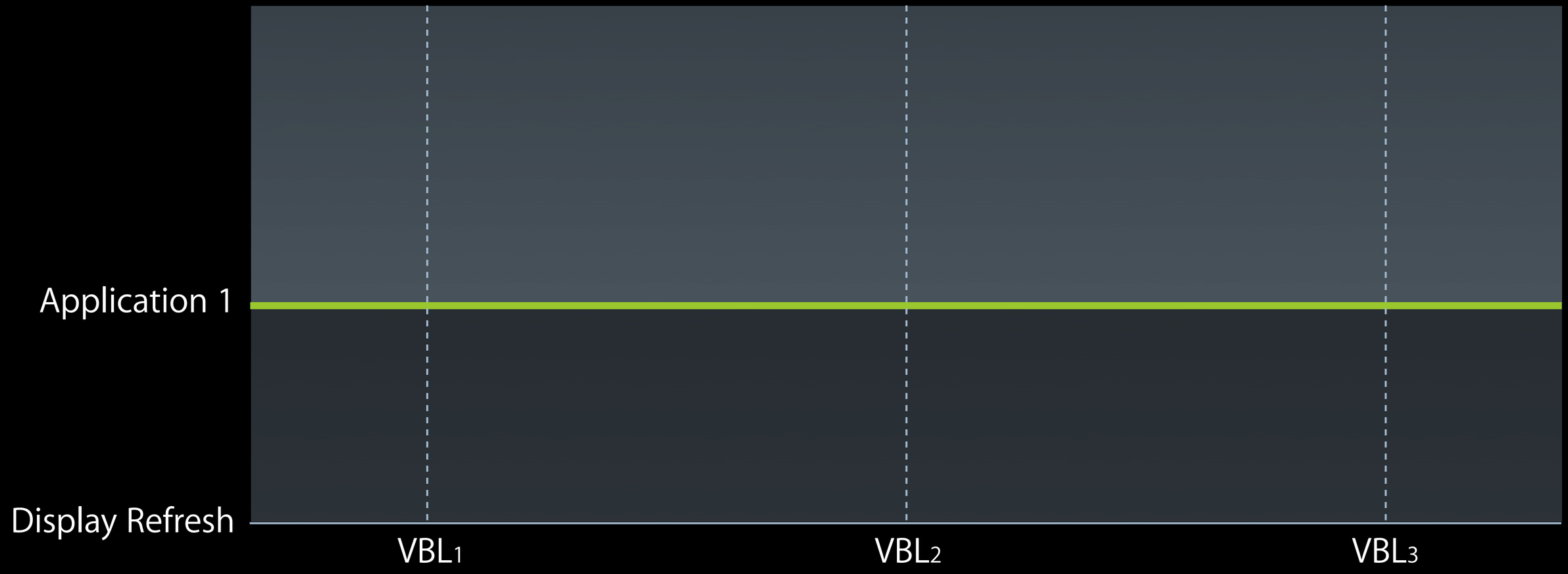
Coalesced updates

http://developer.apple.com/wwdc2005/

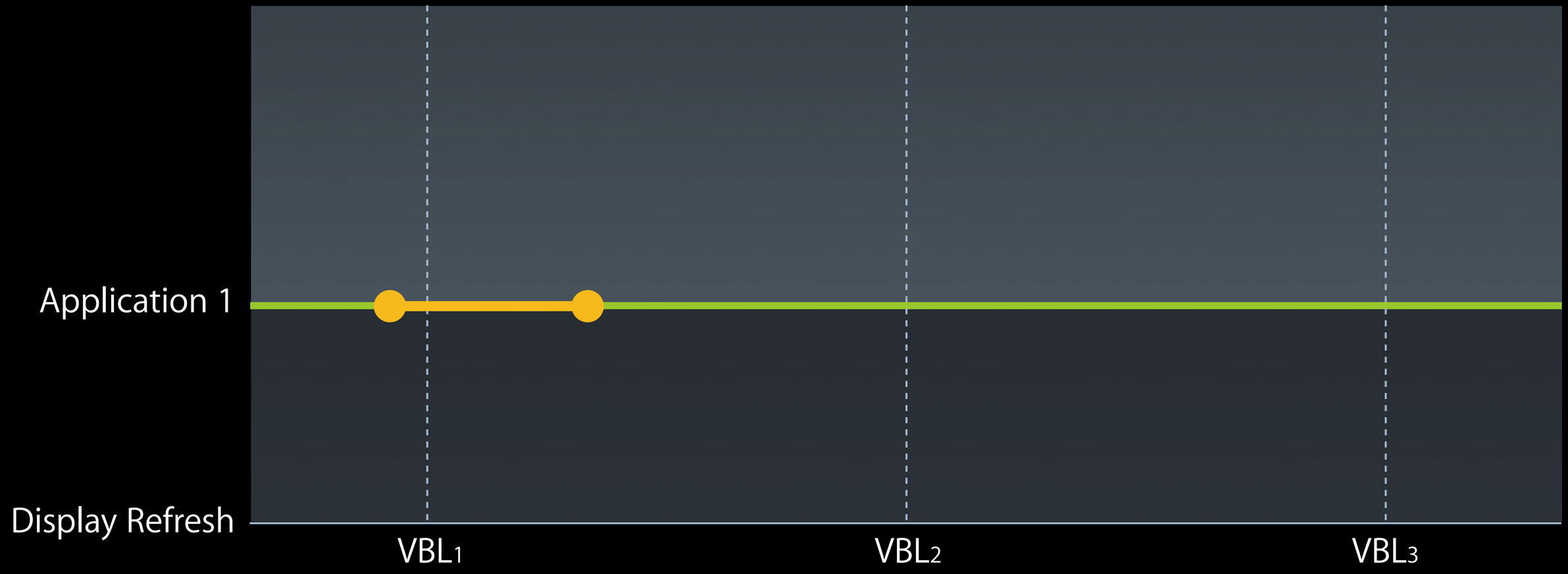
Coalesced Updates

- Guaranteed to be updated at the next vertical blanking interval
- Any area involved in an update will prevent subsequent modification until the update has been committed to the display

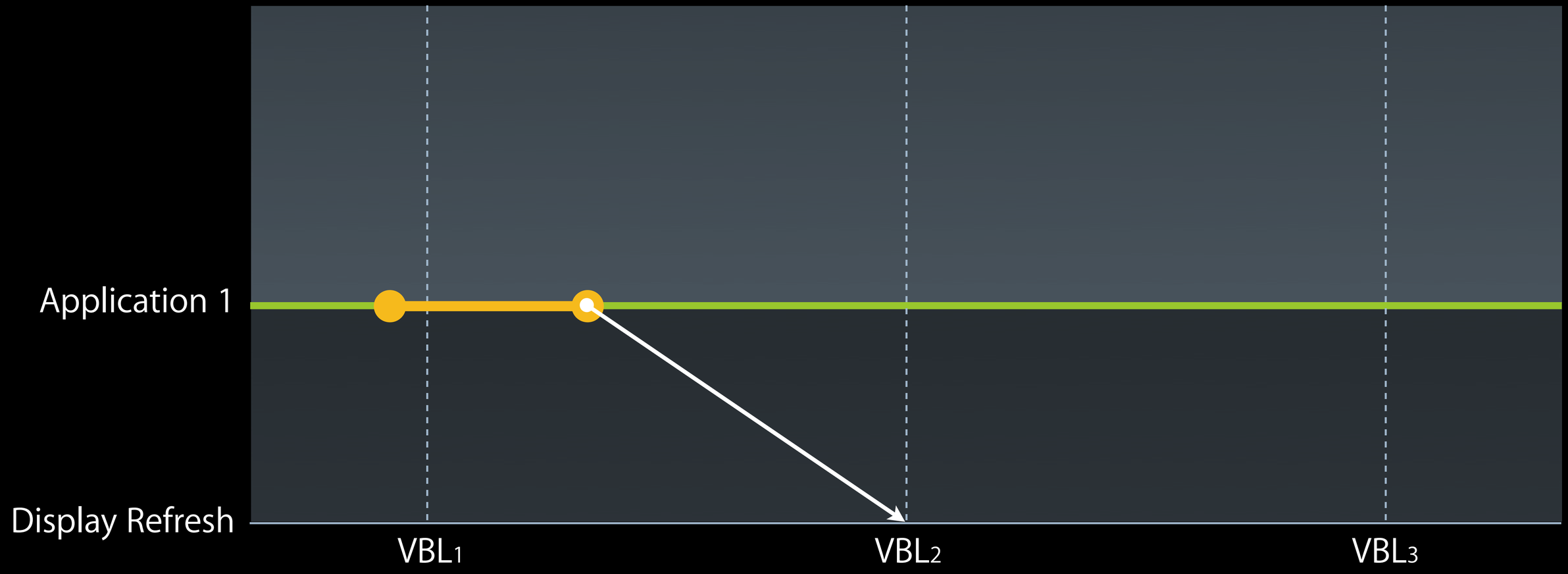
Drawing and Flushing



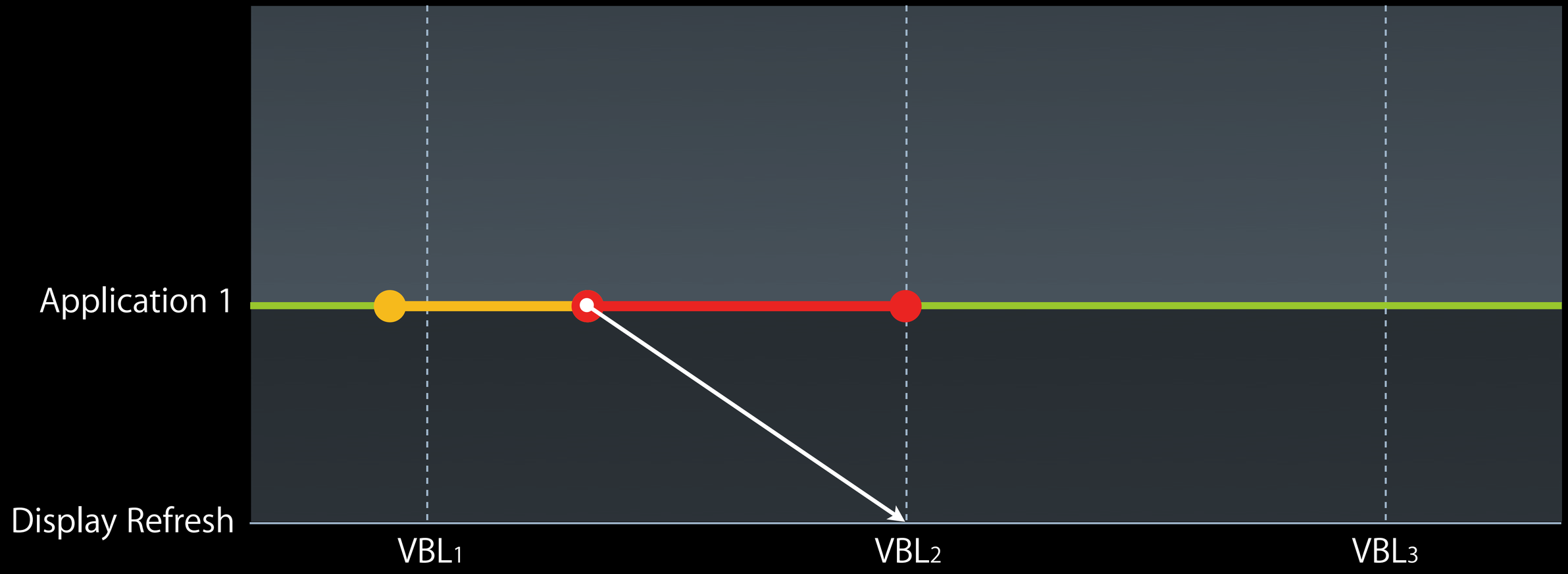
Drawing and Flushing



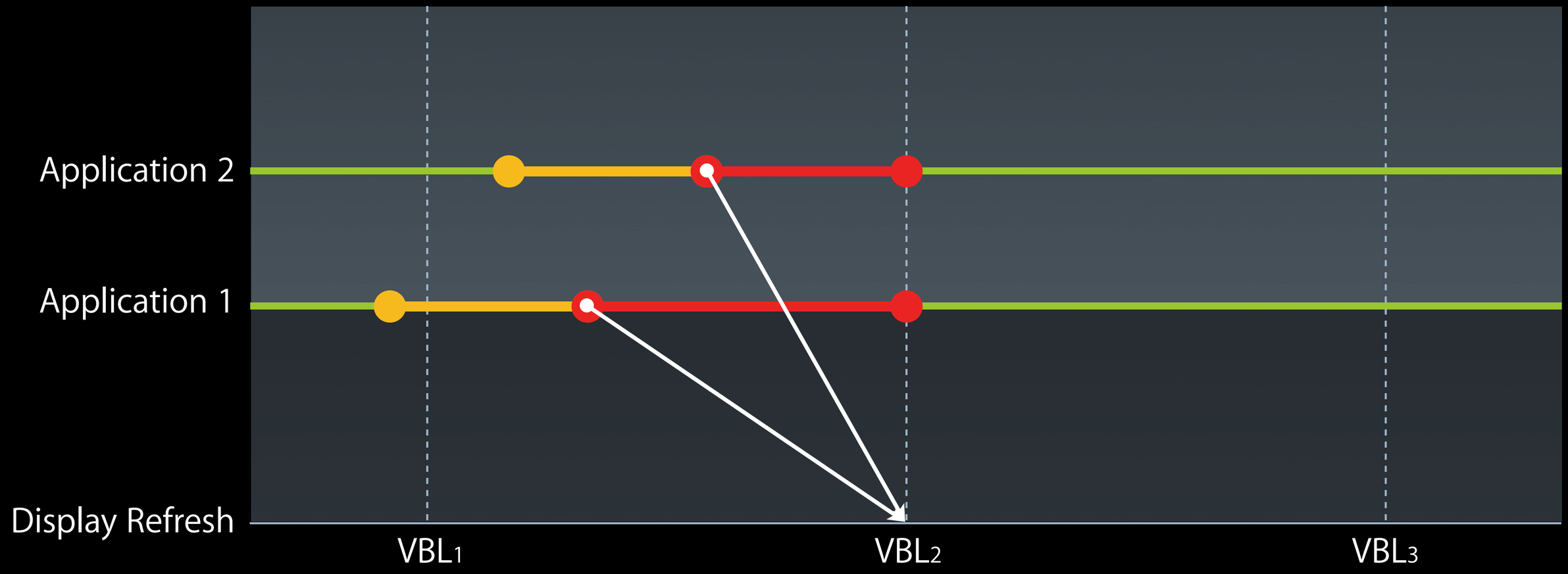
Drawing and Flushing



Drawing and Flushing



Drawing and Flushing



Coalesced Updates

Gotchas

- How to avoid blocking your application
  - Minimize your flushing
  - Do useful work before you start drawing the next frame
  - Scrolling may need special treatment
- How to detect the problem
  - Use Quartz Debug
  - Use Shark in “Time Profile (All Thread States)” mode
- Technote forthcoming

Application Graphics Optimizations

http://developer.apple.com/wwdc2005/

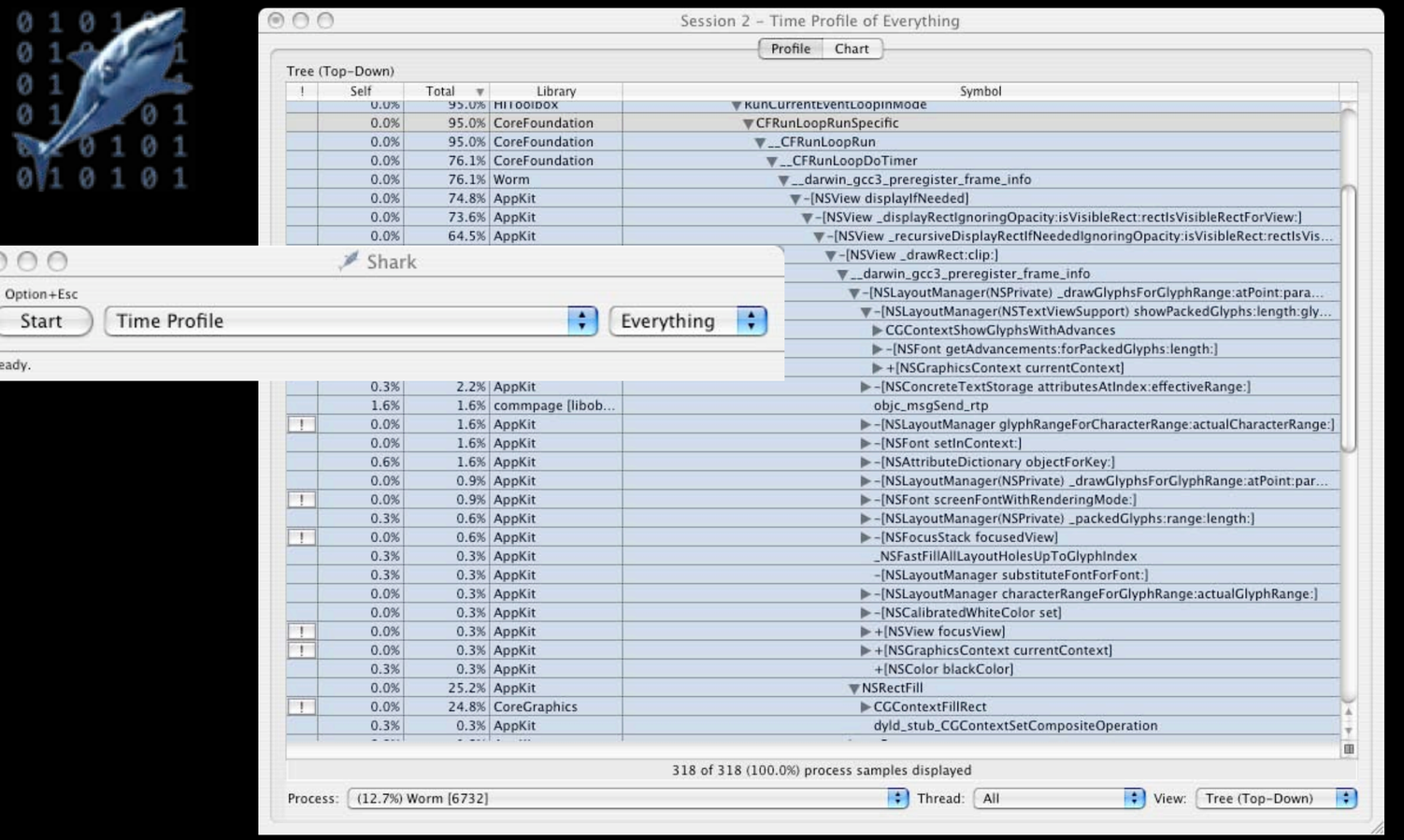
“Premature optimization
is the root of all evil”

Knuth

Optimization Discipline

- Profile your application
- Optimize your application
- Rinse and repeat

Shark



Swimming with Shark

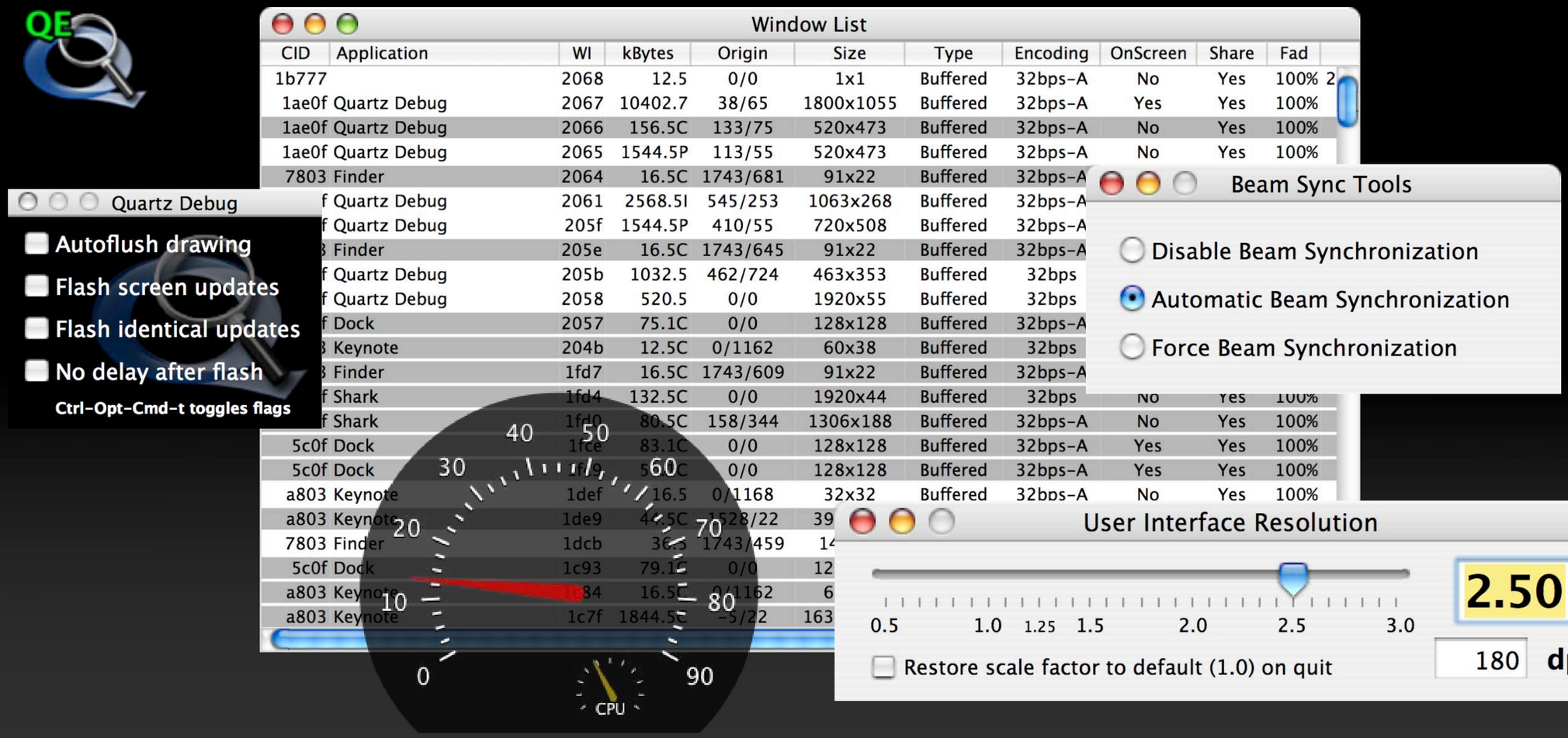
Session 407
Friday 2:00PM

Shark

What to look for

- Time spent in CoreGraphics drawing operations
  - `CGContext[Draw|Fill|EOFill|Stroke]Path`
  - `CGContext[Fill|Stroke|Clear]Rects`
  - `CGContextDrawImage`
  - `CGContextShowGlyphsWithAdvances`
- Flushing is asynchronous
  - Time spent in next drawing operation after flush
    - Typically, `CGContextFillRect` or `CGContextDrawImage`
  - Use Shark's "Time Profile (All Thread States)"
  - Over-flushing will also result in time spent in `CGContextFlush`
- Anything with Lock or Synchronize means contention

Quartz Debug



Quartz Debug

What to look for

- Auto-flush drawing
  - What's being drawn
  - How frequently or redundantly it's being drawn
- Flash screen updates
  - What's being flushed
- Flash identical updates
  - What may not need to be flushed
- Tools gives you more information

Live Resizing

- Consider periodic updates
- Consider drawing lower quality results
- Text layout is time consuming
  - Cache your layout and update periodically
- Avoid network or disk access
  - Don’t block your application during resizes
  - Do the expensive work once the resize has completed
  - And even then, don’t block your UI!

Live Resizing

Carbon

- Only draw or invalidate the newly exposed region
  - The difference between the `kEventParamPreviousBounds` and `kEventParamCurrentBounds` parameters of the the `kEventWindowBoundsChanging` event
  - Handle `kEventControlInvalidateForSizeChange` to perform optimal invalidation
- Notifications are available
  - `kEventWindowResizeStarted`
  - `kEventWindowResizeCompleted`

Live Resizing

Cocoa

- Only draw the newly exposed region
  - `getRectsExposedDuringLiveResize:count:`
- Consider switching to more efficient, but less accurate drawing
  - `inLiveResize`
- Consider preserving content
  - `preservesContentDuringLiveResize`
- Notifications are available
  - `viewWillStartLiveResize`
  - `viewDidEndLiveResize`

Documentation, Sample Code, and Other Resources

http://developer.apple.com/wwdc2005/

Related Sessions

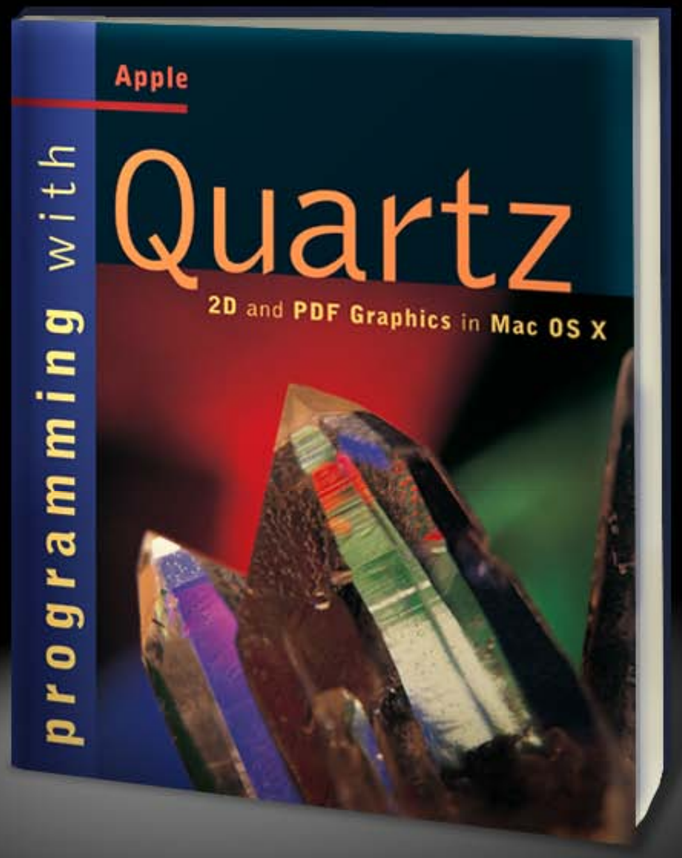
Swimming with Shark	Session 407 Friday 2:00PM

Who to Contact

Travis Brown
Graphics and Imaging Evangelist
travis@apple.com

Additional Resources

- Programming with Quartz
  - 2D and PDF Graphics in Mac OS X
  - Free Chapter on Quartz Performance and Debugging available for download



http://books.elsevier.com/mk/?isbn=0123694736

Q&A

Travis Brown
Graphics and Imaging Evangelist
travis@apple.com

http://developer.apple.com/wwdc2005/

WWDC2005

Worldwide Developers Conference